LEGO® MINDSTORMS® Education EV3

Coding Activities

Second Edition

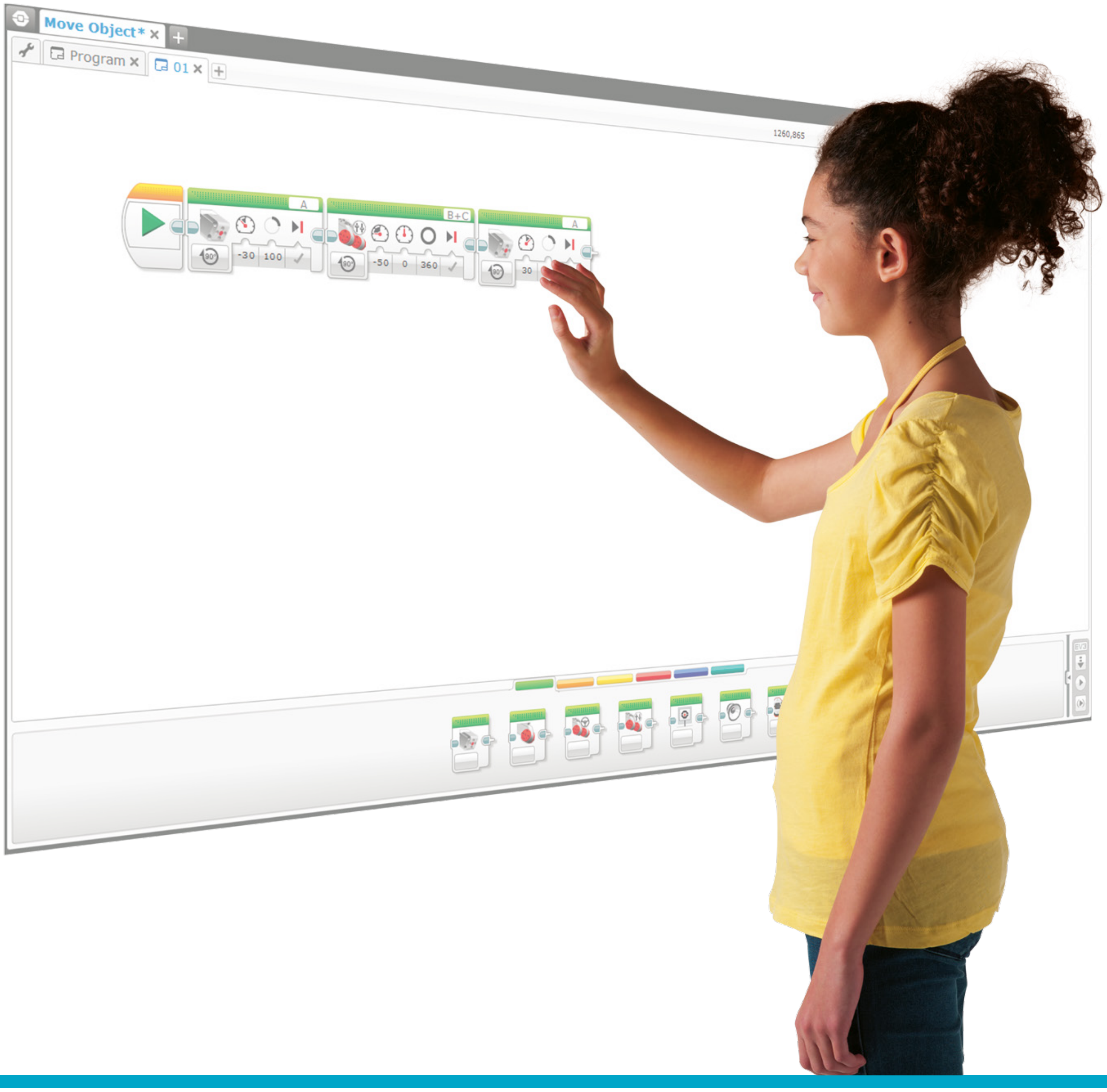
LEGOeducation.com/MINDSTORMS

Table of Content

Introduction

LEGO® Education has developed this materials in order to help teachers facilitate exciting classroom lessons designed around the relevant technology topic of driverless vehicles.

Our intention is to enable students to develop their computational thinking skills as they program solutions in a real-world context.



Target Group

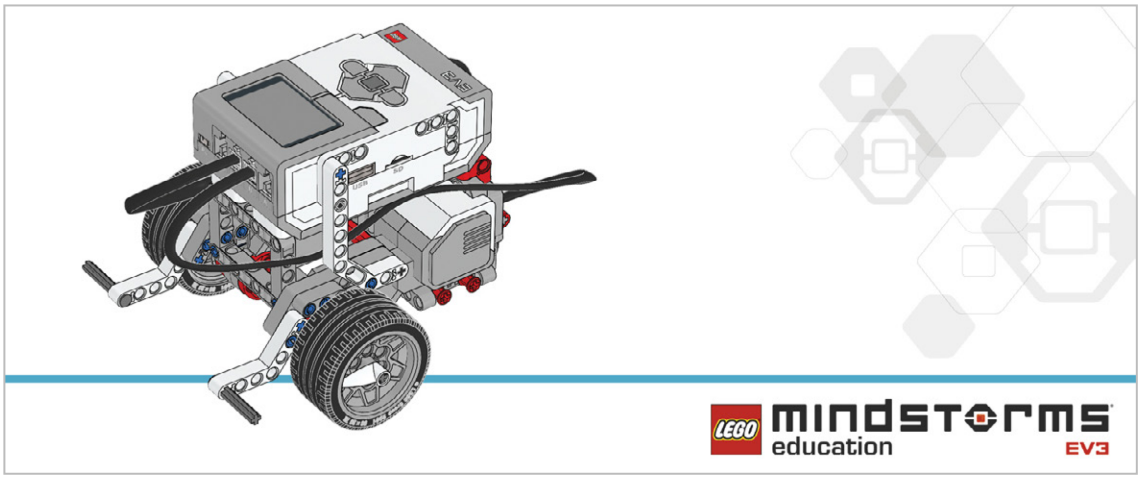
This Material will help teachers to introduce their students to computer programming concepts using a mixture of direct teaching, exploration, and tutorials from the LEGO® MINDSTORMS® Education EV3 Software and Programming app.

This material is aimed at students in grades 6-8, but it can easily be adapted for older students.

Structure of the Lessons

These lessons has been primarily written to address key programming concepts through the exploration of real life problems linked to the theme of autonomous cars. Over the course of these lessons, students will develop their design and computational thinking skills. This material includes many cross-curricular opportunities that can impact other subject areas such as science, mathematics, and engineering design technology.

All lessons are based on the Robot Educator base model and its extensions. Any driving base design can be used. It is a good idea to provide students with many different building experiences in order to develop their design skills.



Each lesson includes a discussion about the structure and design of the program.

Robot Educator Tutorials

Each lesson will require the students to have a good understanding of the EV3 Software or Programming app. This can be achieved by having them complete the basic Robot Educator tutorials. This, supported by direct teaching and exploration, will ensure that students gain the skills and understanding necessary to carry out most tasks given in the lessons.

Activity Flow

There are many ways in which to use the EV3 Coding Activities in your classroom. The material have been organized based on an increasing level of complexity in the programming concepts covered. However, you may organize them in whichever way best suits your classroom needs. You also have the freedom to adapt the lessons to fit the the time you have available.

Each lesson will last approximately:

- 60 minutes, using a pre-built robot with a focus solely on programming
- 90 minutes (2 x 45 minutes), including building, documenting, and sharing
- 135 minutes (3 x 45 minutes), including building, documenting, sharing, and a text-based program comparison.

Text-Based Programming

For this set of activities, LEGO® Education has opted to use ROBOTC as an example of text based programming language. You may choose to use other EV3 compatible text-based programming languages. This material is not intended to teach teachers and students how to use a text-based language, but rather how EV3 programs are the visual equivalent to a text-based solution. For everything you need to know about ROBOTC, go to: http://www.robotc.net/

Where possible, the ROBOTC programs will emulate the EV3 Programs exactly. However, due to the nature of using two different programming languages there will be slight inherent differences. Any differences that have been identified will be highlighted in the relevant ROBOTC program using green text.

Ev3 Programming App

Each of the EV3 Coding Activities can be solved using the EV3 Software or the EV3 Programming App (a tablet environment with only basic programming capabilities). The following table indicates which lessons can be solved using the example code provided with each programming resource.

Lesson	Autonomous Parking	Reversing Safely	Automatic Headlights	Line Detection	Object Detection	Unlocking a Car	Cruise Control	Roaming Vehicles
EV3 Software	X	X	X	X	X	X	X	X
EV3 Programming App	X	X	X	X	X			X
Robot C	X	X	X	X	X	X	X	X

The Variable Block and Array Block required to complete lessons Unlocking a Car and Cruise Control are not yet available in the EV3 Programming App.

Science, Computational Thinking, Coding

While the science and engineering fields originated in the early ages of humankind, computer science has a much younger history. Nevertheless, this young discipline has influenced not only the way in which we approach science and engineering, but also how we live our lives.

Computer Science is a STEM discipline, sharing attributes with science, technology, engineering, and mathematics. All STEM disciplines present opportunities for students to develop a mindset and a lifelong set of practices. Among these practices are the ability to ask questions, to design solutions, and to communicate results.

Computational thinking is another one of these practices. It is a way in which we can think and it is a way in which everybody can solve problems. Computational thinking can be described as a group of skills, one of which is algorithmic thinking. “Code” or “coding” can be used to describe the action of creating an algorithm.

Coding is therefore one vehicle by which to develop students’ computational thinking within a STEM context.

STEM Disciplines

Science, Technology, Engineering,
Mathematics, Computer Science

Develop a minset and life long set of practices

1. Ask questions and solve problems.
2. Use models.
3. Design prototypes.
4. Investigate.
5. Analyze and interpret data.
6. Use computational thinking.

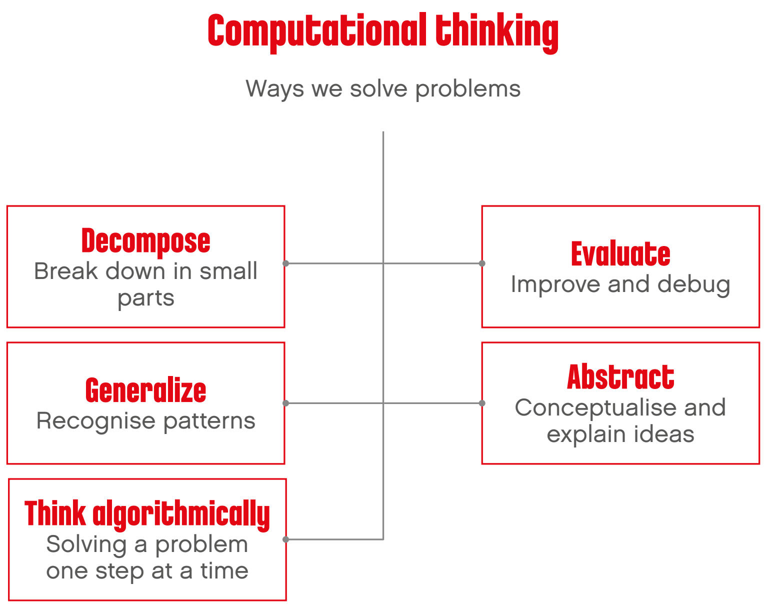
a. Decompose
b. Abstract
c. Think algorythmically (code)
d. Evaluate
e. Generalize

7. Engage in argument from evidence.
8. Obtain, evaluate, and communicate information.

What is Computational Thinking?

The expression “computational thinking” was first used by Seymour Papert, but Professor Jeannette Wing is known to have popularized the idea. She defined computational thinking as:
”the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.” (Wing, 2011)

Computational thinking is used in various fields and situations, and we use it in our daily lives. Computational thinking skills are present in science, engineering, and mathematics. These skills can be defined as the following:



Decomposition

Decomposition is the ability to simplify a problem into smaller parts in order to ease the process of finding a solution. By doing so, the problem becomes easier to explain to another person, or to separate into tasks. Decomposition frequently leads to Generalization.

Example: When going on vacation, the preparation (or project) can be separated into subtasks: booking the airfare, reserving a hotel, packing a suitcase, etc.

Generalization (Pattern Recognition)

Generalization is the ability to recognize the parts of a task that are known, or have been seen somewhere else. This frequently leads to easier ways of designing algorithms.

Example: Traffic lights work by repeating the same series of actions forever.

Algorithmic Thinking (Coding)

Algorithmic Thinking is the ability to create an ordered series of steps with the purpose of solving a problem.
Example one: when we cook from a recipe, we are following a series of steps in order to prepare a meal.
Example two: when using computers, we can code a sequence of actions that tell the computer what to do.

Evaluating or Debugging

This is the ability to verify whether or not a prototype works as intended, and if not, the ability to identify what needs to be improved. It is also the process a computer programmer goes through in order to find and correct mistakes within a program.
Example one: when we are cooking, we will periodically taste the dish to check whether or not it is seasoned correctly.
Example two: when we look for spelling mistakes and missing punctuation in our written work, we are debugging it so that it can be read correctly.

Abstraction

Abstraction is the ability to explain a problem or a solution by removing unimportant details. In other words, being able to conceptualize an idea.
Example: When describing a bicycle, we use only some details to describe it. We might mention its type and color, and add more details for someone who has a real interest in bikes.

A Process for Developing Computational Thinking Skills

Using an Engineering Design Process

When looking for solutions to a problem, engineers use a design process. They go through a series of phases that guide them toward a solution. During each of these phases, some of their skills are used or developed. It is those skills that we we refer to as “computational thinking skills”.

The students will follow a similar process as they complete the EV3 Coding Activities:

Defining the Problem

Students are presented with a topic that guides them to a problem or to a situation they wish to improve. Sometimes, a problem can can be very detailed. To make it easier to solve, the problem can be broken down into smaller parts. By defining the problem in a simple way and by identifying some success criteria, students will develop a skill called “Decomposition”.

In other words:
- Is the student able to explain the problem by themselves?
- is the student able to describe how they will evaluate whether or not they were successful in solving the problem?
- Is the student able to break down the problem into smaller and more manageable parts?

Planning

Students should spend some time imagining different solutions to the problem, and then make a detailed plan for executing one of their ideas. They will define the steps they will need to go through in order to reach the solution. By identifying the parts of the task they might have seen before, they will develop a skill called “Generalization”.

In other words:
- Is the student able to make a list of actions to program?
- Is the student able to identify parts of existing programs that they could use?
- Is the student able to reuse parts of programs?

Trying

Each student is then tasked with creating the final version of their solution. In this phase of the process, they use iconic programming language to activate their LEGO® models. As the students code their ideas, they develop their Algorithmic Thinking skills.

In other words:
- Is the student able to program a solution to a problem?
- Is the student able to use sequence, loops, conditional statements, etc.?

Modifying

Students will evaluate their solution according to whether or not their program and model meet the success criteria. Using their evaluation skills, they will determine whether they need to change, fix, debug, or improve some part of their program.

In other words:
- Is the student making iterations of their program?
- Is the student fixing problems in their program ?
- Is the student able to judge if the solution is linked to the problem ?

Communicating

Students will present the final version of their solution to the class, explaining how their solution meets the success criteria. By explaining their solution in the right level of detail, they will develop their Abstraction and communication skills.

In other words:

- Is the student explaining the most important part of their solution?
- Is the student giving enough detail to enhance comprehension?
- Is the student making sure to explain how their solution meets the success criteria?

Developing Computational Thinking through Coding

In order to develop their Algorithmic Thinking, students will be introduced to some programming principles. As they develop their solutions, they will organize a series of actions and structures that will bring their models to life.

The most common programming principles students will use are:

1. Output

Output is something that is controlled by the program students are writing. Examples of outputs for MINDSTORMS are sounds, lights, display, and turning motors on and off.

2. Input

Input is information that a computer or device receives. It can be entered through the use of sensors in the form of a numeric or text value. For example, a sensor that detects or measures something (such as distance) converts that value into a digital input signal so it can be used in a program.

3. Events (Wait For)

Students can tell their program to wait for something to happen before continuing to execute the sequence of actions. Programs can wait for a specific length of time, or wait for something to be detected by a sensor.

4. Loop

Students can program actions to be repeated, either forever or for a specific amount.

5. Functions

Functions are a group of actions that are to be used together in specific situations. For example, the group of blocks that can be used to make a light blink would together be called, “the blink function”.

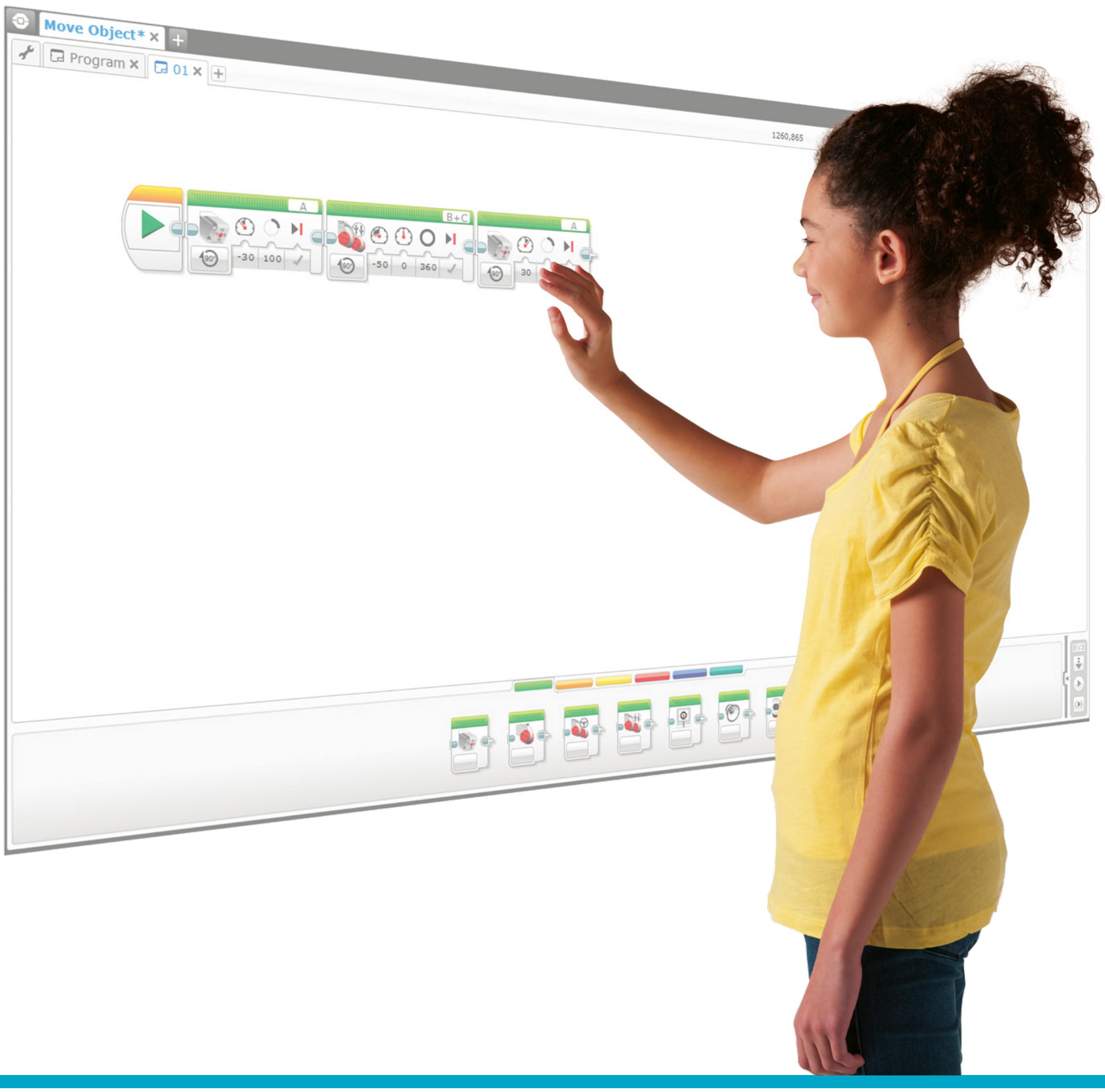
6. Conditions

Conditions are used by students in order to program actions that are to be executed only under certain circumstances. Creating conditions within a program means that some part of the program will never be executed if the condition is never met. For example, if the Touch Sensor is pressed, the motor will start, and if the sensor is released, the motor will stop.

Curriculum Links

In the US and many other countries, education and industry leaders are calling for the inclusion of more programming (coding) experiences in K-12 classrooms. The Computer Science Teachers Association (CSTA) established a set of Computer Science Standards in 2011 and revisited them in 2016.

With this in mind, LEGO® Education has produced this set of lessons using LEGO MINDSTORMS Education EV3 to help students in grades 5-8 tackle this abstract subject.



Lesson	Learning Objectives	Programming concept	Major EV3 Programming Blocks Covered
Autonomous Parking	– Understand that algorithms are capable of carrying out a series of instructions – Explore the concept of Outputs, comparing different ways in which a wheeled robot can move	– Design cars that can park themselves safely without driver intervention	– Move Tank – Wait – Touch Sensor – Brick Status Light
Reversing Safely	– Extend the use of algorithms are capable of carrying out a series of instructions – Extend understanding of outputs	– Design features for a car that will improve safety as it reverses	– Move Steering – Wait – Brick Button – Sound – Display
Automatic Headlights	– Explore the concept of Inputs and the way to control them – Explore the concept of a Wait for function	– Design car features that will improve nighttime driving safety	– Wait – Color Sensor – Loop – Loop Interrupt
Line Detection	– Explore the concept of the Loop – Understand the concept of a switch and how to use it for true and false operations	– Design ways to improve driving safety by helping to prevent drivers from falling asleep and causing an accident	– Wait – Color Sensor – Loop – Switch – Loop Interrupt
Object Detection	– Extend understanding of the Loop	– Design ways to avoid accidents between vehicles and objects in the road	– Wait – Ultrasonic Sensor – Loop – Switch
Unlocking a Car	– Understand simple Boolean logic (such as AND, OR and NOT) and some of its uses in circuits and programming – Use several inputs in combination	– Design a way to use passcodes to protect cars from thieves	– Ultrasonic Sensor – Brick Buttons – Logic – Switch – Loop
Cruise Control	– Use the Variable Block to store information – Develop multi-level programs – Create function blocks (My Blocks)	– Design a cruise control program to assist drivers by making their driving experience less stressful	– Touch Sensor – Loop – Switch – Variable – Math – My Blocks
Roaming Vehicles	– Make appropriate use of data structures such as lists, tables and arrays – Extend Boolean logic and some of its uses in circuits and programming – Use the Variable Block to store information – Use the Array Operations Block	– Design an autonomous car that is safe enough to drive on the streets	– Variable – Brick Buttons – Loop – Array Operations – My Blocks
Project	– Integrate and reinvest what they have learned in the previous coding lessons – Design, use, and evaluate solutions to a real-world problems and physical systems	– Design an autonomous car that can safely cross an intersection	– All of the above

Curriculum Grid

NGSS

● = addresses standard
◐ = partially addresses standard

NGSS		Autonomous Parking	Reversing Safely	Automatic Headlights	Color and Line Detection	Object Detection	Unlocking a Car	Cruise Control	Roaming Vehicles
MS.Engineering Design									
MS-ETS1-1.	Define the criteria and constraints of a design problem, asking questions and defining problems with sufficient precision to ensure a successful solution, taking into account relevant scientific principles and potential impacts on people and the natural environment that may limit possible solutions	●	●	●	●	●	●	●	●
MS-ETS1-2.	Evaluate competing design solutions using a systematic process to determine how well they meet the criteria and constraints of the problem	●	●	●	●	●	●	●	●
MS-ETS1-3.	Analyze data from tests to determine similarities and differences among several design solutions to identify the best characteristics of each that can be combined into a new solution to better meet the criteria for success	●	●	●	●	●	●	●	●
MS-ETS1-4.	Develop a model to generate data for iterative testing and modification of a proposed object, tool, or process such that an optimal design can be achieved			●	●	●	●	●	●

Science and Engineering Practices in the NGSS									
Practice 1	Asking questions (for science) and defining problems (for engineering)	●	●	●	●	●	●	●	●
Practice 2	Developing and using models	●	●	●	●	●	●	●	●
Practice 3	Planning and carrying out investigations								
Practice 4	Analyzing and interpreting data								
Practice 5	Using mathematics and computational thinking	●	●	●	●	●	●	●	●
Practice 6	Constructing explanations (for science) and designing solutions (for engineering)	●	●	●	●	●	●	●	●
Practice 7	Engaging in argument from evidence								
Practice 8	Obtaining, evaluating, and communicating information	●	●	●	●	●	●	●	●

Curriculum Grid

CSTA Standards (2016)

⬣ = addresses standard
◐ = partially addresses standard

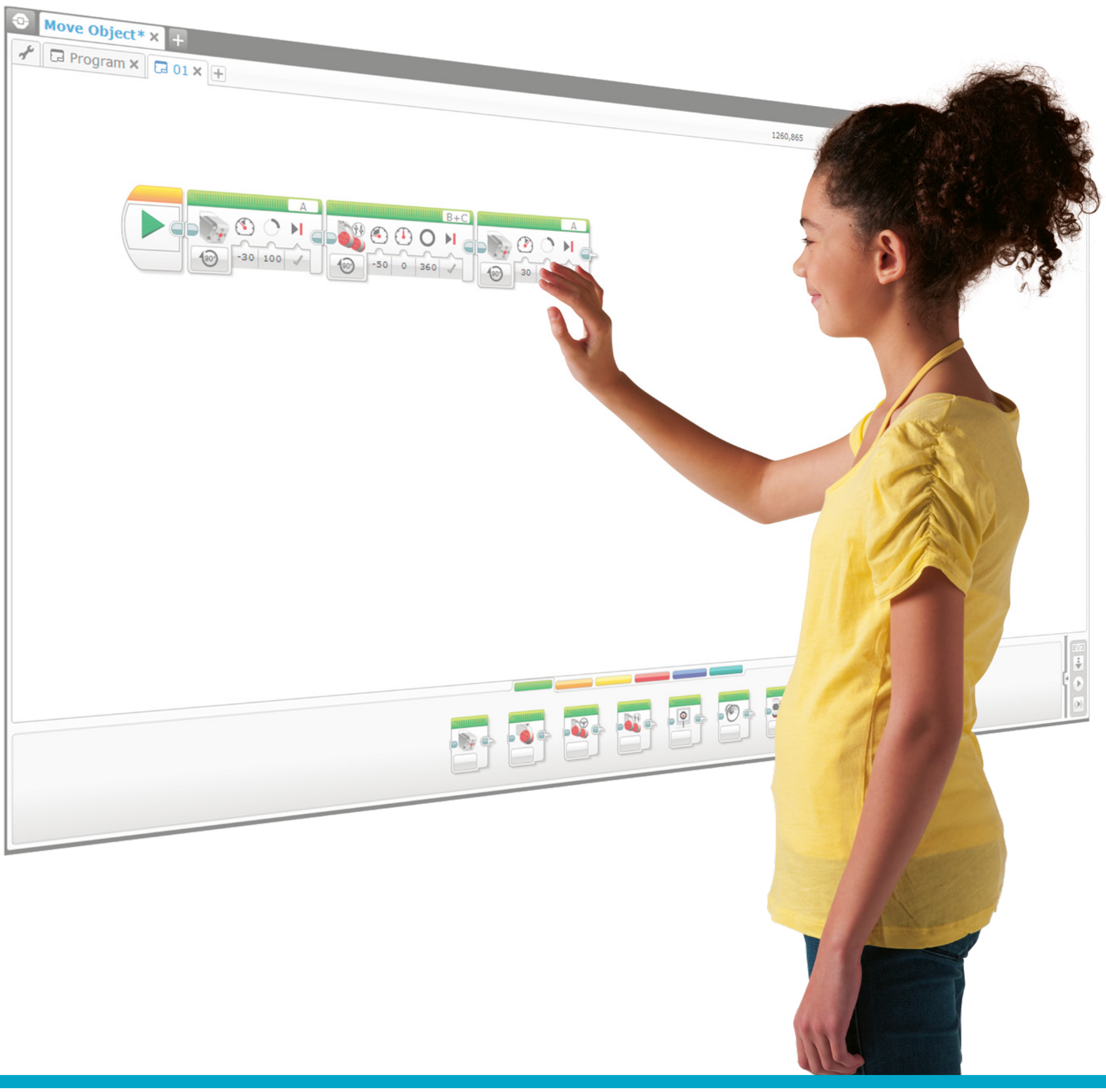
		Autonomous Parking	Reversing Safely	Automatic Headlights	Color and Line Detection	Object Detection	Unlocking a Car	Cruise Control	Roaming Vehicles
Algorithms and Programming									
2-A-2-1	Solicit and integrate peer feedback as appropriate to develop or refine a program	⬣	⬣	⬣	⬣	⬣	⬣	⬣	⬣
2-A-7-2	Compare different algorithms that may be used to solve the same problem in terms of their speed, clarity, and size (e.g., different algorithms solve the same problem, but one might be faster than the other). (Clarification: students are not expected to quantify these differences)	⬣	⬣	⬣	⬣	⬣	⬣	⬣	⬣
2-A-7-3	Provide proper attribution when code is borrowed or built upon	⬣	⬣	⬣	⬣	⬣	⬣	⬣	⬣
2-A-7-4	Interpret the flow of execution of algorithms and predict their outcomes. (Clarification: algorithms can be expressed using natural language, flow and control diagrams, comments within code, and pseudocode)	⬣	⬣	⬣	⬣	⬣	⬣	⬣	⬣
2-A-5-5	Design, develop, and present computational artifacts such as mobile applications that address social problems both independently and collaboratively	⬣	⬣	⬣	⬣	⬣	⬣	⬣	⬣
2-A-5-6	Develop programs, both independently and collaboratively, that include sequences with nested loops and multiple branches. (Clarification: at this level, students may use block-based and/or text-based programming languages)			⬣	⬣	⬣	⬣	⬣	⬣
2-A-5-7	Create variables that represent different types of data, and manipulate their values						⬣	⬣	⬣
2-A-4-8	Define and use procedures that hide the complexity of a task and can be reused to solve similar tasks. (Clarification: students use and modify, but do not necessarily create, procedures with parameters)								⬣
2-A-3-9	Decompose a problem into parts and create solutions for each part	⬣	⬣	⬣	⬣	⬣	⬣	⬣	⬣
2-A-6-10	Use an iterative design process (e.g., define the problem, generate ideas, build, test, and improve solutions) to solve problems, both independently and collaboratively	⬣	⬣	⬣	⬣	⬣	⬣	⬣	⬣
Computing Systems									
2-C-7-11	Justify the hardware and software chosen to accomplish a task (e.g., comparison of the features of a tablet vs. desktop, selecting which sensors and platform to use in building a robot or developing a mobile app)								
2-C-4-12	Analyze the relationship between a device's computational components and its capabilities. (Clarification: computing systems include not only computers, but also cars, microwaves, smartphones, traffic lights, and flash drives)	⬣	⬣	⬣	⬣	⬣	⬣	⬣	⬣
2-C-6-13	Use a systematic process to identify the source of a problem within individual and connected devices (e.g., follow a troubleshooting flow diagram, make changes to software to see if hardware will work, restart device, check connections, swap in working components)	◐	◐	◐	◐	◐	◐	◐	◐

Curriculum Grid

CSTA Standards (2016) ⬢ = addresses standard ◐ = partially addresses standard		Autonomous Parking	Reversing Safely	Automatic Headlights	Color and Line Detection	Object Detection	Unlocking a Car	Cruise Control	Roaming Vehicles
Data and Analysis									
2-D-7-14	Describe how different formats of stored data represent tradeoffs between quality and size. (Clarification: compare examples of music, text, and/or image formats)								
2-D-7-15	Explain the processes used to collect, transform, and analyze data to solve a problem using computational tools (e.g., use an app or spreadsheet form to collect data, decide which data to use or ignore, and choose a visualization method)	◐	◐	◐	◐	◐	◐	◐	◐
2-D-5-16	Revise computational models to more accurately reflect real-world systems (e.g., ecosystems, epidemics, spread of ideas)	⬢	⬢	⬢	⬢	⬢	⬢	⬢	⬢
2-D-4-17	Represent data using different encoding schemes (e.g., binary, Unicode, Morse code, shorthand, student-created codes)								
Impacts of Computing									
2-I-7-18	Summarize negative and positive impacts of using data and information to categorize people, predict behavior, and make recommendations based on those predictions (e.g., customizing search results or targeted advertising based on previous browsing history can save search time and limit options at the same time)								
2-I-7-19	Explain how computer science fosters innovation and enhances nearly all careers and disciplines	◐	◐	◐	◐	◐	◐	◐	◐
2-I-1-20	Provide examples of how computational artifacts and devices impact health and wellbeing, both positively and negatively	⬢	⬢	⬢	⬢	⬢	⬢	⬢	⬢
2-I-1-21	Describe ways in which the Internet impacts global communication and collaborating								
2-I-1-22	Describe ethical issues that relate to computing devices and networks (e.g., equity of access, security, hacking, intellectual property, copyright, Creative Commons licensing, and plagiarism)								
2-I-6-23	Redesign a computational artifact to remove barriers to universal access (e.g., using captions on images, high contrast colors, and/or larger font sizes)								◐
Networks and the Internet									
2-N-7-24	Summarize security risks associated with weak passwords, lack of encryption, insecure transactions, and persistence of data						◐		
2-N-7-25	Simulate how information is transmitted as packets through multiple devices over the Internet and networks						◐		

Assessment

There are many ways in which you can monitor and assess your students' progress through a lesson. This section includes an observation rubrics grid that you can use to provide feedback to your students about the development of their computational thinking skills.



STUDENT-Led Assessment

Encourage your students to tell their learning story. Provide them with the opportunity to share their thinking, ideas, and reflections using the documentation tool(s) they have available. Students can document their thoughts using text, videos, images, sketchnotes, or another creative medium. Allow each student to select the tool(s) they find most appropriate for capturing and sharing their unique thinking, creations, and learning process.

Teacher-Led Assessment

Developing students' science, engineering, and computational thinking skills requires time and feedback. Just as in the design cycle, in which students should understand that failure is part of the process, assessment should provide feedback in terms of what students did well and where they can improve. Problem-oriented learning is not about succeeding or failing. It is about being an active learner and continually building upon and testing ideas.

Giving feedback to students in order to help them develop their skills can be done in various ways. Along with the EV3 Coding Activities, we have provided examples of rubrics that can be completed by:
• Observing students' behavior, reactions, and strategies
• Asking the students questions about their thought processes

As students often work in groups, you can give feedback both on a team level and on an individual level.

Observation Rubrics

Examples of computational thinking rubrics have been provided on the next pages. For every student, or every team, you can use the observation rubrics grid to:
• Evaluate student performance at each step of the process
• Provide constructive feedback to help each student progress

The rubrics are based on these progressive stages:

1. Bronze (Emerging)

The student is at the beginning stages of development in terms of content knowledge, ability to understand and apply content, and/or ability to demonstrate coherent thoughts about a given topic.

2. Silver (Developing)

The student is able to present basic knowledge only (e.g., vocabulary), and cannot yet apply content knowledge, or demonstrate comprehension of the concepts being presented.

3. Gold (Proficient)

The student has concrete levels of comprehension of the content and concepts, and can adequately demonstrate the topics, content, or concepts being taught. The ability to discuss and apply this knowledge outside the required assignment is lacking.

4. Platinum (Accomplished)

The student can take concepts and ideas to the next level, apply concepts to other situations, and synthesize, apply, and extend their knowledge to discussions that include extensions of ideas.

Decomposition Rubrics

	Bronze	Silver	Gold	Platinum	Notes
Describe the problem in your own words.	The student is unable to describe the problem in their own words.	With prompting, the student is able to describe the problem in their own words.	The student is able to describe the problem in their own words.	The student is able to describe the problem in their own words and starts to decompose the problem into smaller parts.	
Describe how you will know whether or not you have found a successful solution to the problem.	The student is unable to describe success criteria.	With prompting, the student is able to describe success criteria.	The student is able to describe success criteria.	The student is able to describe success criteria in detail.	
Describe how you can break the problem down into smaller parts.	The student is unable to break down the problem.	With prompting, the student is able to break down the problem into smaller parts.	The student is able to break down the problem into smaller parts.	The student is able to break down the problem into smaller parts and can describe the links between each of the parts.	

Generalization Rubrics

	Bronze	Silver	Gold	Platinum	Notes
Describe which program you have used from the Program Library (or elsewhere) and why.	The student is unable to describe which program has been used and why.	With prompting, the student is able to identify which program has been used.	The student is able to describe which program has been used and why.	The student is able to describe, in detail, which program has been used and what modifications have been made to it.	
Observe how your students recognize patterns, or reuse concepts they have seen before.	The student is unable to recognize patterns, or reuse concepts seen before.	With prompting, the student is able to recognize patterns, or reuse concepts seen before.	The student is able to recognize patterns, or reuse concepts seen before.	The student is able to recognize patterns, or reuse concepts of their own.	

Algorithmic Thinking Rubrics

	Bronze	Silver	Gold	Platinum	Notes
Describe the list of actions to program.	The student is unable to describe a list of actions.	With prompting, the student is able to describe a list of actions.	The student is able to describe a list of actions.	The student is able to create a detailed list of actions to help them develop their program.	
Describe how you have programmed your solution.	The student is unable to describe the program.	With prompting, the student is able to describe the program.	The student is able to describe the program.	The student is able to describe the program, providing extensive details about each component.	
Describe the programming principles used in your solution (e.g., output, inputs, events, loops, etc.).	The student is unable to describe the programming principles used in their solution.	With prompting, the student is able to describe the programming principles used in their solution.	The student is able to describe the programming principles used in their solution.	The student is able to describe, with extensive comprehension, the programming principles used in their solution.	

Evaluation Rubrics

	Bronze	Silver	Gold	Platinum	Notes
Describe what happened when you executed your program, and whether or not it was what you expected.	The student cannot describe what happened.	With prompting, the student is able to describe what happened, and compare it to what was expected.	The student is able to describe what happened, and compare it to what was expected.	The student is able to describe what happened, compare it to what was expected, and is already finding solutions.	
Describe how you have fixed the problems in your program.	The student cannot describe how they have fixed the problems.	With prompting, the student can describe how they have fixed the problems.	The student can describe how they have fixed the problems.	The student can describe, in extensive detail, how they have fixed the problems.	
Describe how your solution is linked to the problem.	The student is unable to describe how their solution is linked to the problem.	With prompting, the student is able to describe how their solution is linked to the problem.	The student is able to describe how their solution is linked to the problem.	The student is able to describe, in extensive detail, how their solution is linked to the problem.	
Describe some of the different ways in which you have tried to solve the problem.	The student is unable to describe the different ways in which they have tried to solve the problem.	With prompting, the student is able to describe the different ways in which they have tried to solve the problem.	The student is able to describe the different ways in which they have tried to solve the problem.	The student is able to describe the different ways in which they have tried to solve the problem and can explain why each of the options wasn't viable.	

Abstraction Rubrics

	Bronze	Silver	Gold	Platinum	Notes
Describe the most important part of your solution.	The student is not able to describe any parts of their solution.	With prompting, the student is able to describe their solution.	The student is able to describe their solution.	The student is able to describe their solution, focusing on the most important part of the solution.	
Describe the most important details of your solution.	The student is not able to provide any details about their solution.	With prompting, the student is able to provide details about their solution.	The student is able to discuss details of their solution, but some of the details are not essential.	The student is able to discuss the most important details of their solution.	
Describe how your solution met the initial design criteria.	Their student is unable to describe how their solution met the initial design criteria.	With prompting, the student is able to describe how their solution met the initial design criteria.	The student is able to describe how their solution met the initial design criteria.	The student is able to describe, with extraordinary clarity, how their solution met the initial design criteria.	

Autonomous Parking

Design cars that can park themselves safely without driver intervention.

Learning Objectives

Students will:

Understand that algorithms are capable of carrying out a series of instructions in order
Explore the concept of Outputs by compare different ways in which a wheeled robot can move

Vocabulary

Output, algorithm, pseudocode, debug

Grades

6-8

Subjects

Engineering, STEM, Coding

Duration

45-90 Minutes

Difficulty

Beginner

Standards

NGSS
MS-ETS1-1. / MS-ETS1-2. / MS-ETS1-3. / MS-ETS1-4

CSTA
2-A-2-1 / 2-A-7-1 / 2-A-7-3 / 2-A-7-4 / 2-A-5-5 / 2-A-3-9 / 2-A-6-10 / 2-C-4-12 / 2-D-5-16 / 2-I-1-20

Materials Needed

LEGO® MINDSTORMS Education EV3 core set
LEGO MINDSTORMS EV3 Software or Programming app
ROBOTC software (optional)

Autonomous Parking

CONNECT (5 minutes)

Ignite a classroom discussion around the following questions:

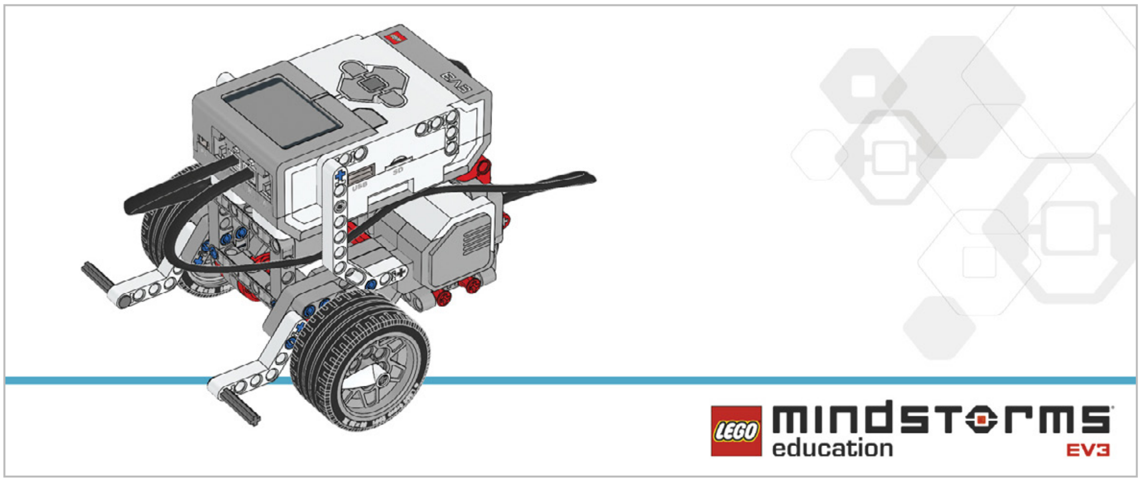
- How do autonomous cars work?
- What would it take to ensure that autonomous cars are safe?
- What types of movements do autonomous cars need to perform?

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their ideas. Encourage them document their thoughts using text, videos, images, sketchnotes, or another creative medium.

CONSTRUCT (15 to 30 minutes)

Build

Students will construct the Robot Educator base model, which is a basic wheeled robot.



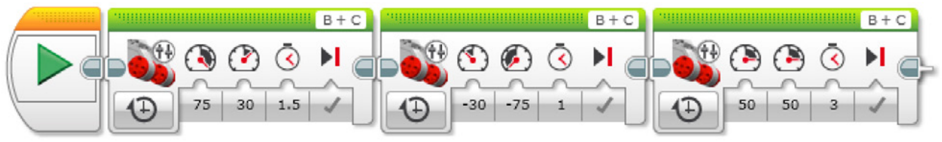
Have the students perform the following building check before they program their robots:

- Are the wires correctly connected from the motors to ports B and C?
- Are the wheels correctly installed?
- Are the wheels rotating freely?

Program

Have the students begin a new project in the EV3 programming environment.
Have them practice by creating a program that will make the robot turn three times.
Encourage the students to explore different ways of making the wheeled robot move. Have them describe the effect(s) of altering the parameters of each of the blocks that they use.

POSSIBLE SOLUTION
FILENAME: CODING-01.EV3 (Tab:1)



THREE POINT TURN

1. Start the Program
2. Turn the driving base and stop after 1.5 seconds.
3. Turn the driving base left and stop after 1 second.
4. Move the driving base forward for 3 seconds.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Note

Refer students to the Robot Educator Tutorials for further assistance.

In the EV3 Software :
Robot Educator > Basics > Straight Move
Robot Educator > Basics > Curved Move
Robot Educator > Basics > Tank Move

In the EV3 Programming App :
Robot Educator > Curved Move

Set up

Before assigning the next task, make sure that you have marked the path the robots must follow and that there is enough space to complete the task. It is a good idea to have students work on a big table or on the floor.

CONTEMPLATE (35 minutes)

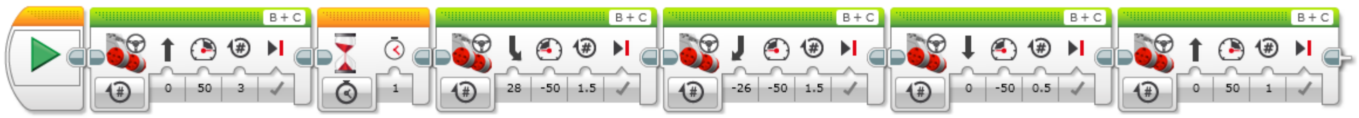
Have the students choose one or all of these autonomous driving scenarios to program:

- Parallel parking
- Angle parking
- Perpendicular parking

They should use a different Programming Canvas for each solution.

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their pseudocode. Encourage them to use text, videos, images, sketchnotes, or another creative medium.

POSSIBLE SOLUTION
FILENAME: CODING-01.EV3 (Tab:2)



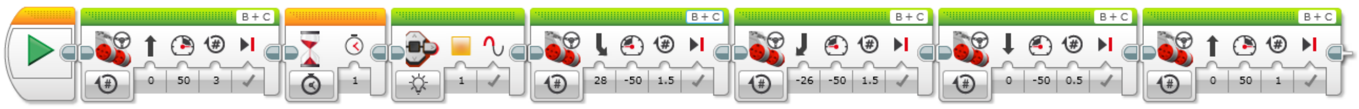
PARALLEL PARKING IN AUTONOMOUS MODE
1. Start the Program
2. Drive forward in a straight line at the desired speed
3. Wait for 1 second
4. Reverse motor rotation while turning for 1,5 rotation
5. Reverse motor rotation while turning the other way for 1,5 rotation
6. Drive backward in a straight line for 0,5 rotation
7. Drive forward in a straight line for 1 rotation

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Differentiation Option

Have the students create a program that simulates appropriate warning lights while parking (e.g., use the EV3 Brick Status Light to display reverse warning lights).

POSSIBLE SOLUTION
FILENAME: CSACTIVITY1.EV3 (Tab:3)



SIMULATING REVERSE GEAR AND REVERSE WARNING LIGHTS
1. Start the Program
2. Drive forward in a straight line at the desired speed
3. Wait for 1 second
4. Turn light on (reverse light)
5. Reverse motor rotation while turning for 1,5 rotation
6. Reverse motor rotation while turning the other way for 1,5 rotation
7. Drive backward in a straight line for 0,5 rotation
8. Drive forward in a straight line for 1 rotation

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Share

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their creations, unique thinking, and learning process. Encourage them to use text, videos, images, sketchnotes, or another creative medium.

Ask one or two groups to demonstrate their programs. Discuss what works well and what could be improved.

Assessment Opportunity
Specific rubrics for assessing computational think can be found under 'Assessment'.

CONTINUE (45 minutes)

Using text-based programming

Have the students explore text-based programming solutions so they can compare different programming languages.

IMPORTANT
The following is a possible solution using the text-based programming language ROBOTC. You may choose to use any other LEGO MINDSTORMS Education EV3 compatible text-based programming languages.

LEGO Education has no ownership of the ROBOTC platform and does not provide any support or guarantee of the quality of the user experience and technology used. All required set up information is provided by ROBOTC at http://www.robotc.net/. We recommend always to reinstall the official LEGO MINDSTORMS EV3 Brick firmware when you finish using other programming languages.

POSSIBLE SOLUTION
FILENAME: CODING-01_1.C (Three Point Turn)

```
#pragma config(StandardModel, "EV3_REMBOT")
/*
Create a three point turn using timing and steering.
*/

task main()
{
  //Turn to the right and stop after 1.5 seconds.
  setMotorSpeed(motorB, 75);
  setMotorSpeed(motorC, 30);
  sleep(1500);

  //Reverse to the left and stop after 1 second.
  setMotorSpeed(motorB, -30);
  setMotorSpeed(motorC, -75);
  sleep(1000);

  //You should now be straight and facing the other way. Drive off.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);
  sleep(3000);
}
```

Program solutions for this lesson are available for download at:
http://www.legoeducation.com

WHAT IS NEXT?

Reversing safely

Autonomous Parking

Student Worksheets

Design cars that can park themselves safely without driver intervention.

Student Worksheets

CONNECT

Make sure that you can answer to the following questions:

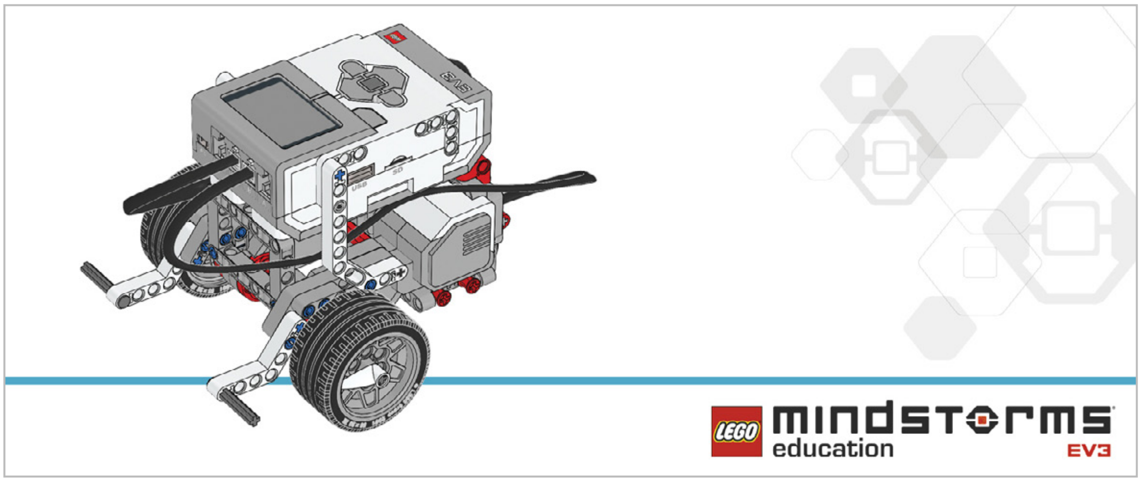
- How do autonomous cars work?
- What would it take to ensure that autonomous cars are safe?
- What types of movements do autonomous cars need to perform?

Think about what you have learned, then document it. Describe the problem in your own words. Creatively record your ideas and findings.

CONSTRUCT

Build

Start by constructing this model.



Program

Write a program that will make the robot turn three times in various ways.

Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas and findings.

CONTEMPLATE

Choose one of the following autonomous driving scenarios and create a program for it:
- Parallel parking
- Angle parking
- Perpendicular parking.

Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas, and findings.

Differentiation

Create a program that simulates displaying appropriate warning lights while parking. (e.g., use the EV3 Brick Status Light to display reverse warning lights).

Share

Think about what you have learned, then document it. Creatively record and present your ideas, creations, and findings.

CONTINUE

Explore text-based programming solutions for this lesson and compare these solutions using different programming languages.

Reversing Safely

Design features for a car that will improve safety as it reverses.

Learning Objectives

Students will:

Extend the use of algorithms are capable of carrying out a series of instructions in order
Extend understanding of outputs

Vocabulary

Input, output, algorithm, pseudocode, debug, wait for

Grades

6-8

Subjects

Engineering, STEM, Coding

Duration

45-90 Minutes

Difficulty

Intermediate

Standards

NGSS
MS-ETS1-1. / MS-ETS1-2. / MS-ETS1-3. / MS-ETS1-4

CSTA
2-A-2-1 / 2-A-7-1 / 2-A-7-3 / 2-A-7-4 / 2-A-5-5 / 2-A-3-9 / 2-A-6-10 / 2-C-4-12 / 2-D-5-16 / 2-I-1-20

Materials Needed

LEGO® MINDSTORMS Education EV3 core set.
LEGO MINDSTORMS EV3 Software or Programming app
ROBOTC software (optional)

Reversing Safely

CONNECT (5 minutes)

In this lesson, students will use the Brick Status Light and Display Block to program their wheeled robot to simulate visual signals for pedestrians, other car drivers, and passengers when reversing. Students will also use the Touch Sensor to simulate forward and reverse gears.
Ignite a classroom discussion around the following questions:

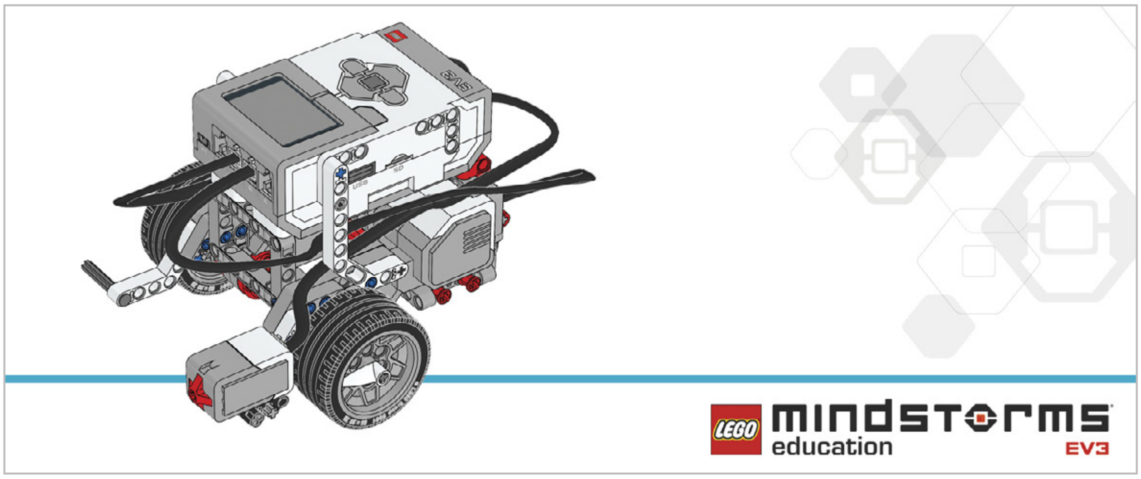
- What are the dangers of reversing a car or other motorized vehicle?
- What are your area's regional statistics concerning accidents with reversing vehicles?
- How can a vehicle reverse more safely?
- What happens on the outside of vehicles when they are reversing to let pedestrians and other drivers know what they are doing?

Ask your students discuss briefly what happens when cars reverse. Discuss how the reverse lights warn other motorists and pedestrians of what the vehicle is doing. Explain that the indicators on the dashboard communicate to the driver and their passengers which gear the car is in and the direction of travel.
Allow the students to select the tool(s) they find most appropriate for capturing and sharing their ideas. Encourage them to document their thoughts using text, videos, images, sketchnotes, or another creative medium.

CONSTRUCT (15 to 30 minutes)

Build

Students will construct the Robot Educator base model, then they will add the Touch Sensor.

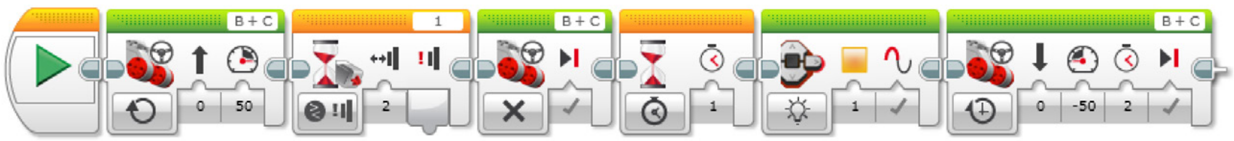


Have the students perform the following building check before they program their robots:
- Are the wires correctly connected from the motors to ports B and C ?
- Are the wheels correctly installed?
- Are the wheels rotating freely?
- Are the wires correctly connected from the Touch Sensor to port 1 ?

Program

Have the students program their robot to simulate a vehicle reversing, including the use of light indicators.
Have them use the Touch Sensor as a bumper or as a "brake" in order to reverse the robot.
Allow the students to select the tool(s) they find most appropriate for capturing and sharing their pseudocode. Encourage them to use text, videos, images, sketchnotes, or another creative medium.

POSSIBLE SOLUTION
FILENAME: CODING-02.EV3 (Tab:1)



BUMP AND REVERSE

1. Start the Program.
2. Drive forward in a straight line at the desired speed.
3. Wait for the Touch Sensor (simulating changing to reverse gear) to be "bumped".
4. Stop the motors.
5. Wait for 1 second.
6. Activate the "reverse lights" (amber lights on EV3 Brick).
7. Reverse direction of travel for 2 seconds.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Note

You can use the EV3 Software to demonstrate the Brick Status Light, and the three different modes of the Touch Sensor (i.e., pressed, released, bumped).
Refer students to the Robot Educator Tutorials for further assistance.

In the EV3 Software :

Robot Educator > Hardware > Brick Status Light
Robot Educator > Hardware > Brick Display
Robot Educator > Hardware > Touch Sensor

Set up

Before assigning the next task, make sure that there is enough space for them to complete the task. It is a good idea to have students work on a big table or on the floor.

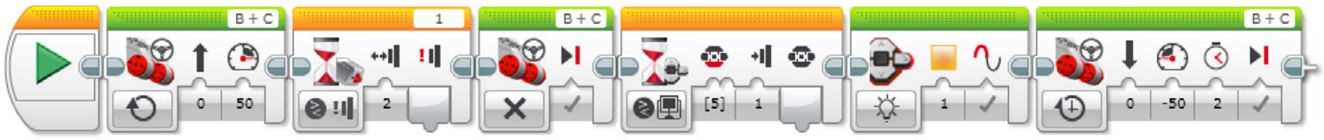
CONTEMPLATE (35 minutes)

Introduce the use of the button on the EV3 Brick to act as the "reverse" gear.
Ask the students how they think the screen could be used as part of an automated car. Discuss ways in which the screen could be used to relay important information to the car's passengers.
Students will build on their previous learning by creating programs that simulate forward and reverse gears in the following ways:

– Use the Touch Sensor as a bumper to stop the robot.
– Use the Brick Buttons to reverse the robot.
– Have the robot indicate its actions.

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their pseudocode. Encourage them to use text, videos, images, sketchnotes, or another creative medium.

POSSIBLE SOLUTION
FILENAME: CODING-02.EV3 (Tab:2)



BUMP, WAIT FOR REVERSE GEAR

1. Start the program.
2. Drive forward in a straight line at the desired speed.
3. Wait for the Touch Sensor (simulating change to reverse gear) to be "bumped".
4. Stop the motors.
5. Wait for the Brick Button (simulating change to reverse gear) to be pressed.
6. Activate the "reverse lights" (amber lights on the EV3 Brick).
7. Reverse the direction of travel for 2 seconds.

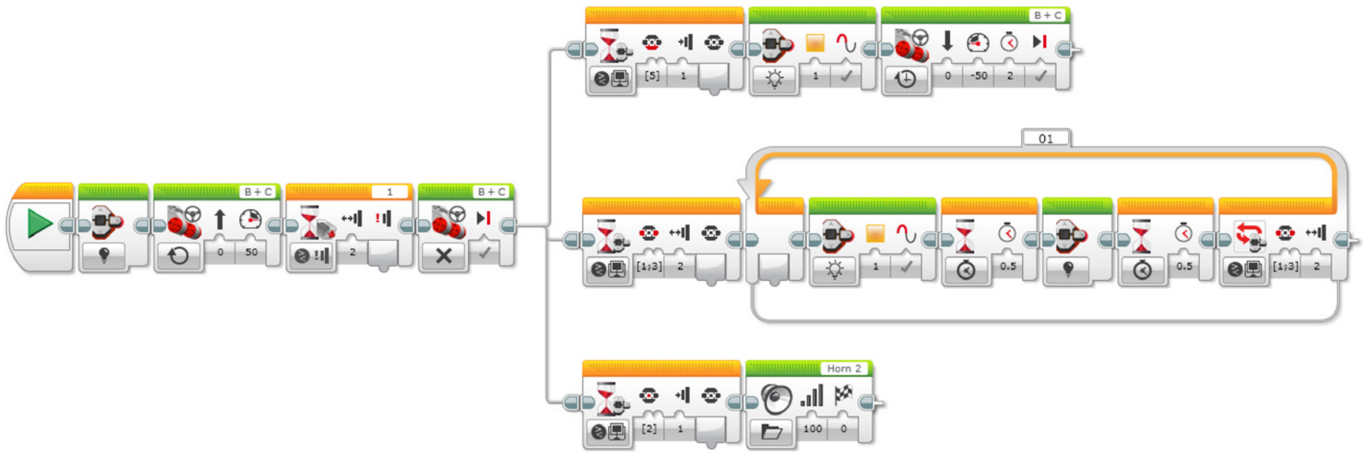
Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Differentiation Option

Have the students utilize the EV3 Brick screen as a dashboard indicator.
They will further develop their programs by using the Display Block to incorporate the EV3 Brick screen so that the forward and reverse motion of the wheeled robot will trigger the screen to display arrows indicating the direction of travel.
Have the students integrate other signals, such as:

– Turn signals (a blinking light when turning left or right)
– Horn

POSSIBLE SOLUTION
FILENAME: CODING-02.EV3 (Tab:3)



BUMP, WAIT FOR REVERSE GEAR WITH INDICATORS
1. Start the program.
2. Turn off lights on the EV3 Brick.
3. Drive forward in a straight line at the desired speed.
4. Wait for the Touch Sensor (simulating change to reverse gear) to be "bumped".
5. Stop the motors and start 3 simultaneous tasks.

TASK 1
6. Wait for the Down Brick Button (simulating change to reverse gear) to be pressed.
7. Activate the "reverse lights" (amber lights on the EV3 Brick).
8. Reverse the direction of travel for 2 seconds.

TASK 2
9. Wait for the Left or Right Brick Button to be "bumped".
10. Activate the amber lights on the EV3 Brick.
11. Wait for 0.5 seconds.
12. Turn off lights on the EV3 Brick.
13. Wait for 0.5 seconds.
14. Repeat steps x to x until the Left or Right Brick Button is "bumped" again.

TASK 3
15. Wait for Enter Brick Button (simulating horn) to be pressed.
16. Play Horn 2 sound.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Share

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their creations, unique thinking, and learning process. Encourage them to use text, videos, images, sketchnotes, or another creative medium.
Ask one or two groups to demonstrate their programs and how their wheeled robot moves.
How many variations did the group as a whole come up with? Compare the many possible solutions to the given problem.

Assessment Opportunity
Specific rubrics for assessing computational think can be found under 'Assessment'.

CONTINUE (45 minutes)

Using text-based programming

Have the students explore text-based programming solutions so they can compare different programming languages.

IMPORTANT
The following is a possible solution using the text-based programming language ROBOTC. You may choose to use any other LEGO MINDSTORMS Education EV3 compatible text-based programming languages.

LEGO Education has no ownership of the ROBOTC platform and does not provide any support or guarantee of the quality of the user experience and technology used. All required set up information is provided by ROBOTC at http://www.robotc.net/. We recommend always to reinstall the official LEGO MINDSTORMS EV3 Brick firmware when you finish using other programming languages.

POSSIBLE SOLUTION
FILENAME: CODING-02_1.C

```
#pragma config(StandardModel, "EV3 _ REMBOT")
/*
Create a program where the robot drives forward until the Touch Sensor
is pressed. The robot then reverses flashing an orange LED.
*/
task main()
{
  //Set the MotorSpeed to 50%.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);
  //Wait for touch sensor to be pressed.
  while(getTouchValue(touchSensor) ==0)
  {
        sleep(10);
  }
  //Set the MotorSpeed to 0% or off. Wait 1 second.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  sleep(1000);
  //Flash the EV3 LED Orange.
  setLEDColor(ledOrangeFlash);
  //Set the MotorSpeed to -50%(reverse). Wait 2 seconds.
  setMotorSpeed(motorB, -50);
  setMotorSpeed(motorC, -50);
  sleep(2000);
}
```

Program solutions for this lesson are available for download at:
http://www.legoeducation.com

WHAT IS NEXT?

Automatic Headlights

Reversing Safely

Student Worksheets

Design features for a car that will improve safety as it reverses.

Student Worksheets

CONNECT

Make sure that you can answer to the following questions:

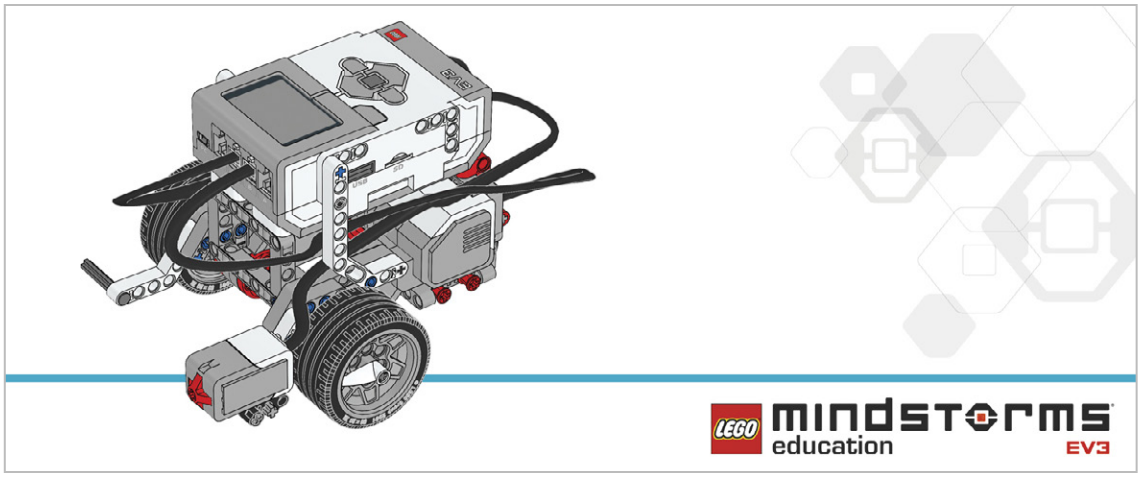
- What are the dangers of reversing a car or other motorized vehicle ?
- What are your area's regional statistics concerning accidents with reversing vehicles?
- How can a vehicle reverse more safely?
- What happens on the outside of vehicles when they are reversing to let pedestrians and other drivers know what they are doing?

Think about what you have learned, then document it. Describe the problem in your own words. Creatively record your ideas and findings.

CONSTRUCT

Build

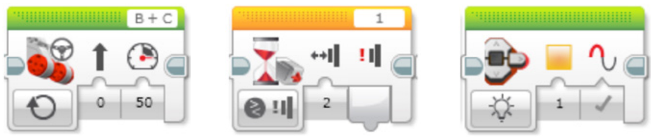
Start by constructing this model.



Program

Write a program that will drive your wheeled robot forward and put it into reverse when you press the Touch Sensor.
Your wheeled robot should display reverse warning lights.
Use the Brick Status Light to simulate reverse warning lights.

Consider using these blocks in your solution:

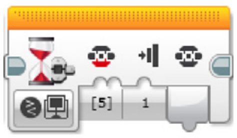


Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas and findings.

CONTEMPLATE

Extend your program so that your wheeled robot has more functions:
- Use the Touch sensor as a bumper to stop the robot.
- Use the Brick Buttons to reverse the robot.
- Have the robot indicate its actions.

Consider using this block in your solution:



Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas, and findings.

Differentiation Option

Integrate new signals to your program, such as:
- Turn signals (a blinking light when turning left or right)
- Horn

Share

Think about what you have learned, then document it. Creatively record and present your ideas, creations, and findings.

Consider the following questions:
How could you improve your program?
Have you used too many blocks? Is there a more efficient way of building your program?

CONTINUE

Explore text-based programming solutions for this lesson and compare these solutions using different programming languages.

Automatic Headlights

Design car features that will improve nighttime driving safety.

Learning Objectives

Students will:

Explore the concept of Inputs and the way to control them
Explore the concept of a Wait for function

Vocabulary

Input, output, algorithm, pseudocode, Wait, Color Sensor, debug, ambient light, loop, Boolean logic, parallel programming

Grades

6-8

Subjects

Engineering, STEM, Coding

Duration

45-90 Minutes

Difficulty

Intermediate

Standards

NGSS
MS-ETS1-1. / MS-ETS1-2. / MS-ETS1-3. /MS-ETS1-4.

CSTA
2-A-2-1 / 2-A-7-2 / 2-A-7-3 / 2-A-7-4 / 2-A-5-5 / 2-A-5-6 / 2-A-3-9 / 2-A-6-10 / 2-C-7-11 / 2-C-4-12 / 2-D-5-16 / 2-I-1-20

Materials Needed

LEGO® MINDSTORMS Education EV3 core set.
LEGO MINDSTORMS EV3 Software or Programming app
ROBOTC software (optional).
Direct light source (e.g., flashlight)

Automatic Headlights

CONNECT (5 minutes)

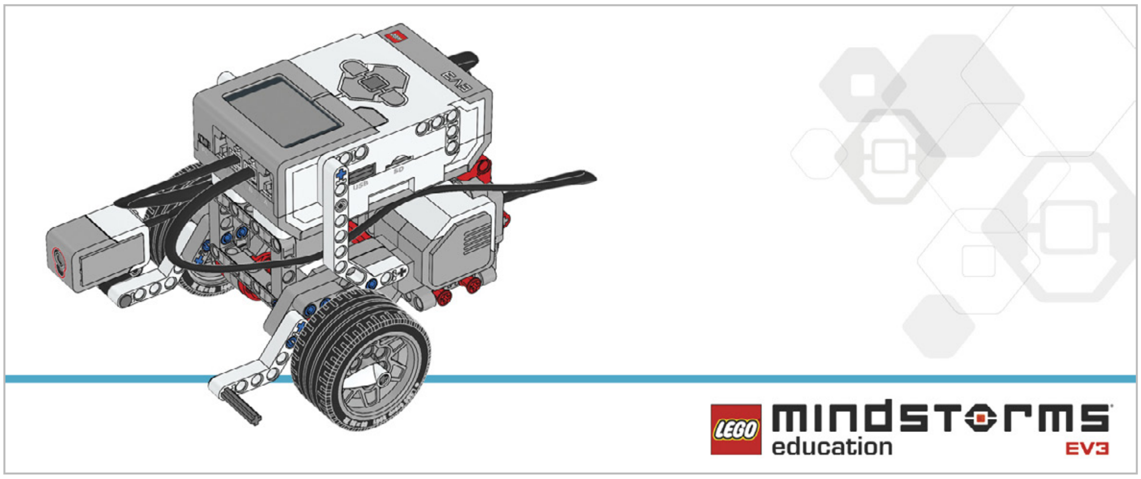
– What are the dangers of driving at night?
– How can we define the concept of ambient light?
– How can drivers be assisted in their nighttime driving?

Tell the students that during this lesson they will be exploring how changes in ambient light can trigger reactions in programs, and that control of a program through a decision is an example of Boolean logic.
Allow the students to select the tool(s) they find most appropriate for capturing and sharing their ideas. Encourage them to document their thoughts using text, videos, images, sketchnotes, or another creative medium.

CONSTRUCT (15 to 30 minutes)

Build

Students will construct the Robot Educator base model, then they will add the Color Sensor pointing forward.



Have the students perform the following building check before they program their robots:
– Are the wires correctly connected from the motors to ports B and C?
– Are the wheels correctly installed?
– Are the wheels rotating freely?
– Are the wires correctly connected from the Color Sensor to 3?

Program

Have the students begin a new project in the EV3 programming environment.
Introduce the Color Sensor and demonstrate to the students that it has three modes:

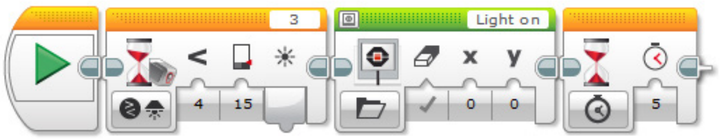
- Color, which detects LEGO® system colors
- Ambient Light Intensity, which detects light levels
- Reflected Light Intensity, which measures the light from the Color Sensor that is reflected off of a nearby surface back to the sensor

Have the students write a program using the Color Sensor in Compare Ambient Light Intensity mode to simulate automatic headlights.
Students can use various methods to simulate the headlights, such as:

- Displaying an image of a light bulb on the EV3 Brick Display
- Using colors of the on the EV3 Brick

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their pseudocode. Encourage them to use text, videos, images, sketchnotes, or another creative medium.

POSSIBLE SOLUTION
FILENAME: CODING-03.EV3 (Tab:1)



AUTONOMOUS AMBIENT LIGHT DETECTION
1. Start the program.
2. Wait for the level of ambient light intensity to drop below 15.
3. Show a light bulb on the EV3 Brick Display.
4. Wait for 5 seconds.

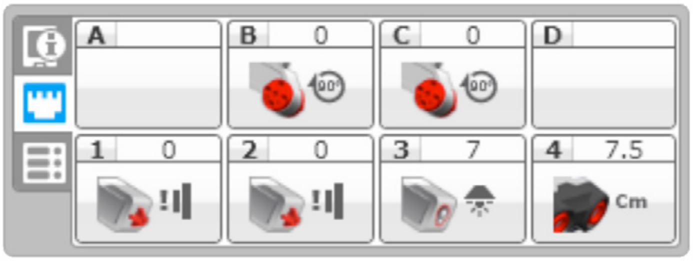
Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Note

Refer students to the Robot Educator Tutorials for further assistance.

In the EV3 Software :
Robot Educator > Basics > Loop
Robot Educator > Hardware > Color Sensor - Light
Robot Educator > Beyond Basics > Multitasking
Robot Educator > Hardware > Brick Display

Show the students how to use the Port View in the EV3 Software to take readings of ambient light (e.g., Port View showing ambient light intensity, port 3).



Explain that they will be exploring how loops work and how to incorporate these into their programs. Explain the concept of a loop and how it can be used to make programs run indefinitely, until manually stopped.

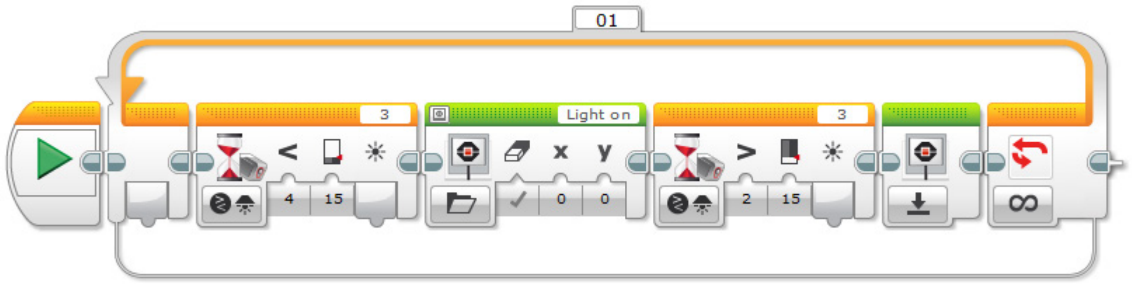
Setup

Have students point a flashlight at the Color Sensor to create a change in ambient light luminosity.

CONTEMPLATE (35 minutes)

Students will create a program that reacts to changes in ambient light intensity, and which simulates automatic headlights more closely. The program will need to turn the lights on when it becomes dark and turn the lights off when it becomes light again.
Highlight that the ambient light intensity settings in their programs will need to be constantly monitored and debugged to ensure that the program is operating correctly.

POSSIBLE SOLUTION
FILENAME: CODING-03.EV3 (Tab:2)



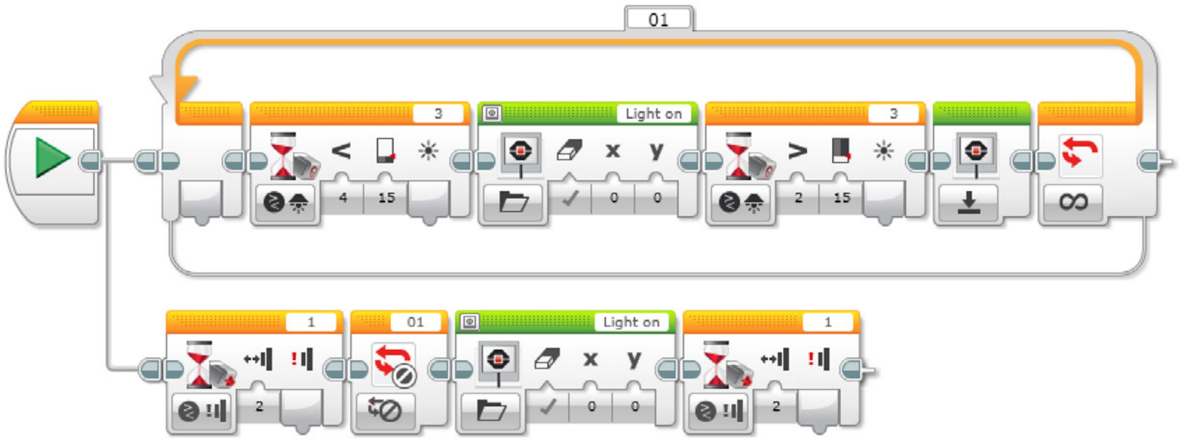
REPEAT OF AUTONOMOUS AMBIENT LIGHT DETECTION
1. Start the program.
2. Wait for the level of ambient light intensity to drop below 15.
3. Show a light bulb on the EV3 Brick Display.
4. Wait for the level of ambient light intensity to rise above 15.
5. Reset the EV3 Brick Display.
6. Repeat steps 2 to 5 forever.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Differentiation Option

Students will use multitasking/parallel programming to simulate automatic and manual light switches.
They can use a Touch Sensor or the Brick Button to simulate the manual switch.
Using the Touch Sensor or the Brick Button as a manual override introduces students to the use of Boolean logic.
Demonstrate to the students how to drag a “wire” from the Start Block in order to create a parallel program (i.e., multitasking).
Show the students the Loop Interrupt Block and explain that we can use this block to stop whatever is happening within the loop in the parallel program.

POSSIBLE SOLUTION
FILENAME: CODING-03.EV3 (Tab:3)



MANUAL CONTROL OVER AUTONOMOUS AMBIANT LIGHT DETECTION
1. Start the program.
TASK 1
2. Wait for the level of ambient light intensity to drop below 15.
3. Show a light bulb on the EV3 Brick Display.
4. Wait for the level of ambient light intensity to rise above 15.
5. Reset the EV3 Brick Display.
6. Repeat steps 2 to 5 of Task 1 forever.
TASK 2
7. Wait for the Touch Sensor (simulating the manual light switch) to be “bumped”.
8. Interrupt the Loop named “01” (override the automatic program).
9. Show a light bulb on the EV3 Brick Display.
10. Wait for the Touch Sensor to be “bumped” again (the light will go off because the program ends).

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Share

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their creations, unique thinking, and learning process. Encourage them to use text, videos, images, sketchnotes, or another creative medium.
Ask one or two groups to demonstrate their programs.
Ask the students to share what they expected to happen vs. what actually happened with their programs.
Ask them whether anything about the results of their programs surprised them.

Assessment Opportunity
Specific rubrics for assessing computational think can be found under 'Assessment'.

CONTINUE (45 minutes)

Using text-based programming

Have the students explore text-based programming solutions so they can compare different programming languages.

IMPORTANT
The following is a possible solution using the text-based programming language ROBOTC. You may choose to use any other LEGO MINDSTORMS Education EV3 compatible text-based programming languages.

LEGO Education has no ownership of the ROBOTC platform and does not provide any support or guarantee of the quality of the user experience and technology used. All required set up information is provided by ROBOTC at http://www.robotc.net/. We recommend always to reinstall the official LEGO MINDSTORMS EV3 Brick firmware when you finish using other programming languages.

POSSIBLE SOLUTION
FILENAME: CODING-03_1.C

```
#pragma config(Sensor, S1, touchSensor, sensorEV3 _ Touch)
#pragma config(Sensor, S4, sonarSensor, sensorEV3 _ Ultrasonic)
#pragma config(Motor, motorB, rightMotor,tmotorEV3 _ Large, PIDControl,
driveRight, encoder)
#pragma config(Motor, motorC, leftMotor, tmotorEV3 _ Large, PIDControl,
driveLeft, encoder)

/*
Create a program that turns on the lights automatically using the Color
Sensor when the ambient light level drops below a certain level.
*/

task main()
{
  //Wait for the level of ambient light intensity to drop below 15.
  while(getColorAmbient(colorSensor) >= 15)
  {
        //Do nothing while we're waiting.
        sleep(1);
  }
  //Display Image "Light on" on the LCD Screen.
  drawBmpfile(0, 127, "Light on");
  //Stay on for 5 seconds.
  sleep(5000);
}
```

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

WHAT IS NEXT?

Line Detection

Automatic Headlights

Student Worksheets

Design car features that will improve nighttime driving safety.

Student Worksheets

CONNECT

Make sure that you can answer the following questions:

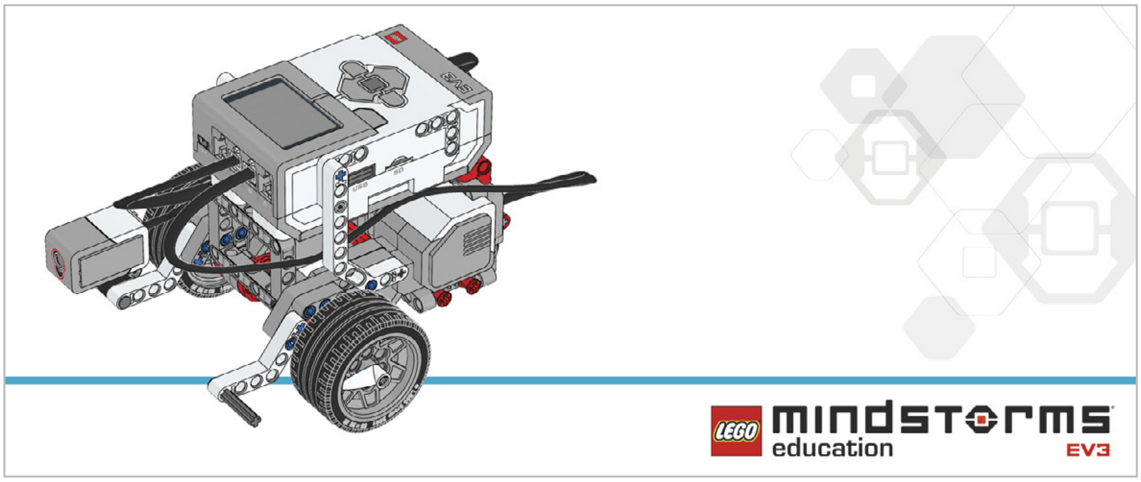
– What are the dangers of driving at night?
– How can we define the concept of ambient light?
– How can drivers be assisted in their nighttime driving?

Think about what you have learned, then document it. Describe the problem in your own words. Creatively record your ideas and findings.

CONSTRUCT

Build

Start by constructing this model.

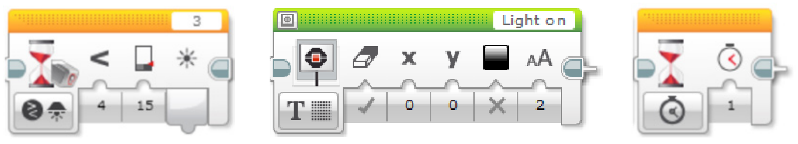


Program

Write a program that will simulate the automatic headlights on a car.

– Find a light bulb image for the EV3 Brick Display that you can incorporate into your program.
– Use the Color Sensor to trigger your “light bulb” to turn on.
– Take ambient light readings from the Port View in order to properly calibrate your program.

Consider using these blocks in your solution:

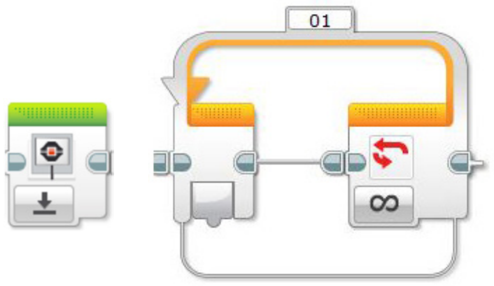


Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas and findings.

CONTEMPLATE

Now that your automatic headlights turn on successfully, you will need to extend your program so that they switch off when it becomes light again. To do this, you will need to create a program that repeats itself so you don't need to keep restarting it.

Consider adding these blocks to your solution:



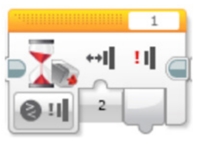
Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas, and findings.

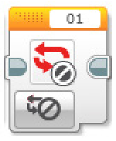
Differentiation option

Create a program that gives more control over your car's automatic headlights so that the lights can be switched on and off manually. Many modern cars have this function, which gives the driver the option to override the automatic program.

– You might need to explore new ways of structuring your program by using parallel programming or multitasking.
– You could use a Touch Sensor to simulate the manual switch.
– You might need to use the Loop Interrupt Block to override the automatic control.

Consider adding these blocks to your solution:





Think about what you have learned, then document it. Creatively record and present your ideas, creations, and findings.

Share

Consider the following questions:
What challenged you? Where there any surprises? How could you improve your program?
Could your program have been more streamlined? Have you used too many blocks? Is there a more efficient way of building your program?
How could your program be used in real-world scenarios?

CONTINUE

Explore text-based programming solutions for this lesson and compare these solutions using different programming languages.

Line Detection

Design ways to improve driving safety by helping to prevent drivers from falling asleep and causing an accident.

Learning Objectives

Students will:

Explore the concept of the Loop
Understand the concept of a switch and how to use it for true and false operations

Vocabulary

Input, output, algorithm, pseudocode , Wait, Color Sensor, debug, ambient light, reflected light, loop, Boolean logic, switch

Grades

6-8

Subjects

Engineering, STEM, Coding

Duration

45-90 Minutes

Difficulty

Intermediate

Standards

NGSS
MS-ETS1-1. / MS-ETS1-2. / MS-ETS1-3. /MS-ETS1-4.

CSTA
2-A-2-1 / 2-A-7-2 / 2-A-7-3 / 2-A-7-4 / 2-A-5-5 / 2-A-5-6 / 2-A-3-9 / 2-A-6-10 / 2-C-7-11 / 2-C-4-12 / 2-D-5-16 / 2-I-1-20

Materials Needed

LEGO® MINDSTORMS Education EV3 core set
LEGO MINDSTORMS EV3 Software or Programming app
ROBOTC software (optional)

Line Detection

CONNECT (5 minutes)

Explain to the students that they will once again be using the Color Sensor. They will extend their understanding of how this sensor reacts to light by using reflected light intensity to create a program that will drive their wheeled robot along a given track.
Tell them that they will use the Color Sensor to make their wheeled robot move more autonomously in order to simulate how an autonomous car might respond to traffic lights.
They will create a program that will make their wheeled robot drive around a given course or track.
Ignite a classroom discussion around the following questions:

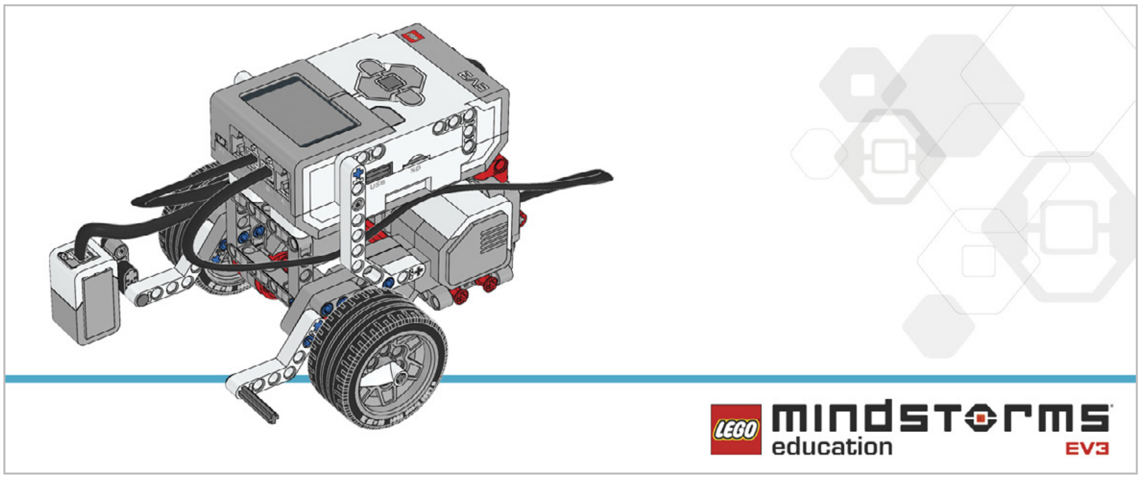
- Can autonomous cars react to different traffic light signals?
- What can happen if a driver falls asleep while driving?
- How can we detect when a driver is falling asleep?

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their ideas. Encourage them to document their thoughts using text, videos, images, sketchnotes, or another creative medium.

CONSTRUCT (15 to 30 minutes)

Build

Students will construct the Robot Educator base model, then they will add the Color Sensor pointing down.



Have the students perform the following building check before they program their robots:

- Are the wheels rotating freely?
- Are the wires correctly connected from the Color Sensor to port 3?

Program

Have the students begin a new project in the EV3 programming environment.
The students will begin exploring the function of the Color Sensor that recognizes LEGO® brick colors by programming their wheeled robot to drive along a path and stop at a red brick.
Have the students simulate a vehicle's behavior at traffic lights by having their wheeled robot respond to a series of green and red signals. Placing their code inside a loop allows for the possibility of multiple "traffic lights" along a track.
Allow the students to select the tool(s) they find most appropriate for capturing and sharing their pseudocode. Encourage them to use text, videos, images, sketchnotes, or another creative medium.

Note

Refer students to the Robot Educator Tutorials for further assistance.

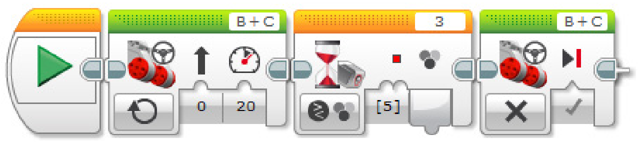
In the EV3 Software :
Robot Educator > Basics > Straight Move
Robot Educator > Basics > Curved Move
Robot Educator > Hardware > Color Sensor - Color
Robot Educator > Beyond Basics > Loop
Robot Educator > Basics > Stop at Line
Robot Educator > Beyond Basics > Switch

- Students will need to use the Wait Block to do this. Point out that the Wait Block can be configured to be triggered by multiple colors, or just one. In this case, students will create a program that uses the Color Sensor to stop the motors when it detects the color red.
- Point out to the students that they will need to make sure all other colors are deselected for the Color Sensor to respond most effectively to the colors they choose (red and green).
- Explain also that they will be exploring how switches work, and how to incorporate these into their programs.
- Explain that they will be exploring how loops work and how to incorporate these into their programs.

Setup

Use the technic beams available in the EV3 core set to simulate green and red lights. Place the beams on the table so the Color Sensor can detect them while rolling over them.

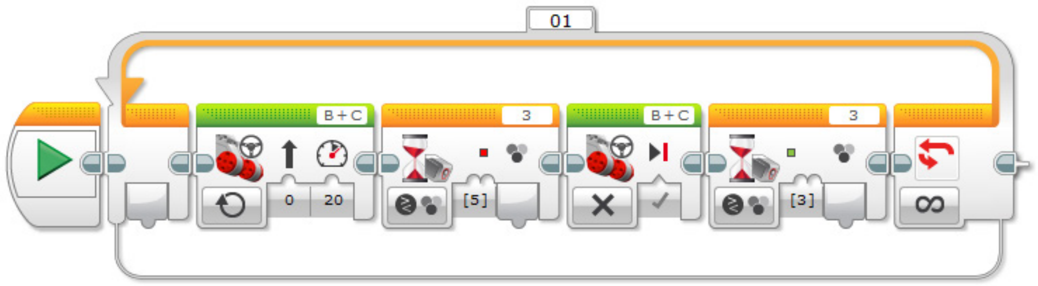
POSSIBLE SOLUTION
FILENAME: CODING-04.EV3 (Tab:1)



RED LIGHT DETECTION
1. Start the program.
2. Start motors B and C (drive forward).
3. Wait for the Color Sensor to detect the color red.
4. Stop the motors.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

POSSIBLE SOLUTION
FILENAME: CODING-04.EV3 (Tab:2)



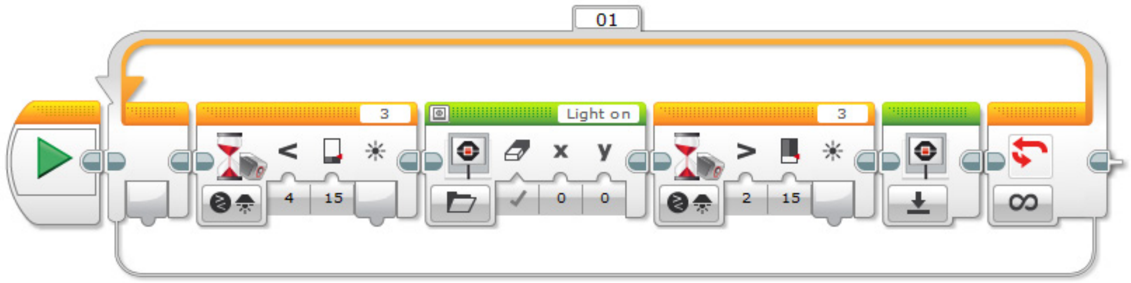
RED AND GREEN LIGHT DETECTION IN LOOP
2. Start motors B and C (drive forward).
3. Wait for the Color Sensor to detect the color red.
4. Stop the motors.
5. Wait for the Color Sensor to detect the color red.
6. Repeat steps 2 to 5 forever.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

CONTEMPLATE (35 minutes)

Students should use the same function of the Color Sensor to recognize when the robot is crossing a line. Use a thick (approx. 2 cm or 3/4 in) dark line if you have a white or light surface, or a white line if you have a dark surface.
Have the students simulate alarm signal for the driver if the robot is crossing this line. This feature often available in new cars.

POSSIBLE SOLUTION
FILENAME: CODING-04.EV3 (Tab:3)



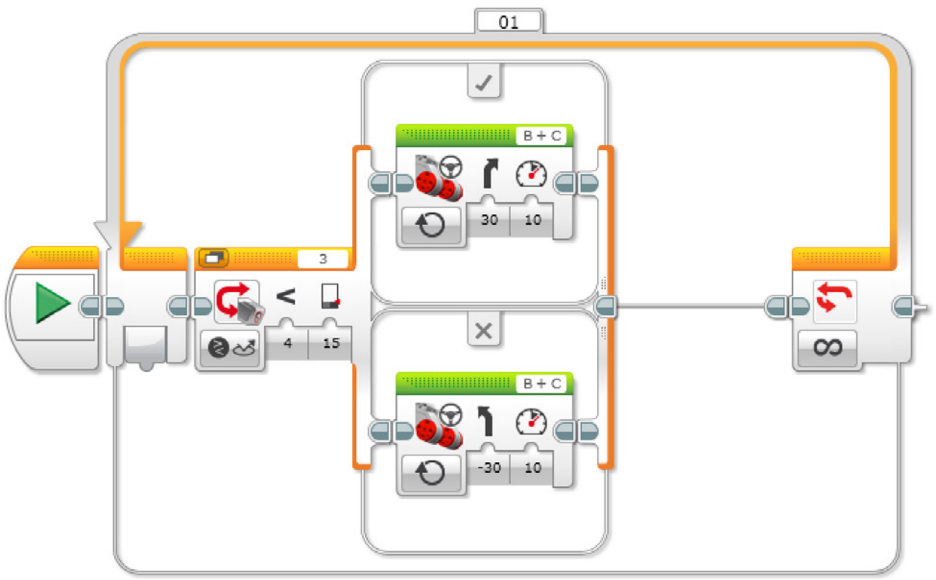
LINE DETECTION IN LOOP
1. Start the program.
2. Start motors B and C (drive forward with a curve toward the line).
3. Wait for the Color Sensor to detect the color black, then start tasks 1 and 2.
TASK 1
4. Play sound "Horn 1".
TASK 2
5. Start motors B and C (drive forward with a curve away from the line).
6. Wait for the Color Sensor to detect the color white.
7. Repeat steps 2 to 6 forever.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Differentiation Option

The students will create an automated, driverless vehicle that can follow a line.
Have the students explore how an automated vehicle might be guided along a road or track.
The students will need to be introduced to the Switch Block, which will operate inside a loop.
Explain that the Switch Block can be used to automate a program that allows the wheeled robot to operate autonomously.
Also explain that the Switch Block can be used to control the flow of a program and that the default Switch Block, using the Touch Sensor, is a classic example of Boolean logic.
Demonstrate how to change the Switch Block to the Color Sensor, and explain that the trigger point is used to create the true/false statement (looking at the Switch Block, note how the program flows above the trigger point to do one thing, or below it do another).
Point out that in order to create the line-following program, they will need to “wiggle” the wheeled robot along the line. In other words, the wheeled robot will turn left and then right depending on whether the line (i.e., trigger) has been crossed. Find a suitable video online to demonstrate an example of this to the students. Point out that the Move Steering Blocks used in this challenge need to be set to “On”, not “On for” (e.g., Seconds, Degrees, or Rotations).
Once the wheeled robot is following the line, can it be improved to behave more like a car (i.e., move in a straight line rather than a wiggle)?

POSSIBLE SOLUTION
FILENAME: CODING-04.EV3 (Tab:4)



LINE FOLLOWING IN LOOP
1. Start the program.
2. Start motors B and C (drive forward with a curve toward the line).
3. Wait for the Color Sensor to detect the color black.
4. Start motors B and C (drive forward with a curve away from the line).
5. Wait for the Color Sensor to detect the color white.
6. Repeat steps 2 to 5 forever.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Note

Students will once again use the Color Sensor, but this time they will need to program it so that it responds to reflected light intensity. They will need to take reflected light intensity readings from the Port View in order to gauge what value to input into the Wait Block.
This will work best using black or blue tape on a very light (or white) surface.
You will need to spend some time explaining the concept of a switch and how it is an example of Boolean logic.
A possible extension from here would be to add a second Color Sensor, and combine the line-follow and traffic light programs to simulate automated passenger services, such as a train system in an airport.

Share

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their creations, unique thinking, and learning process. Encourage them to use text, videos, images, sketchnotes, or another creative medium.
This lesson has covered a lot of new concepts and introduced several new blocks from the EV3 Software. Use this time to recap this information and ensure that the students understand it.
Ask one or two groups to demonstrate their programs.
Ask the students to share what they expected to happen vs. what actually happened with their programs.
Ask them whether anything about the results of their programs surprised them.

Assessment Opportunity

Specific rubrics for assessing computational think can be found under 'Assessment'.

CONTINUE (45 minutes)

Using text-based programming

Have the students explore text-based programming solutions so they can compare different programming languages.

IMPORTANT

The following is a possible solution using the text-based programming language ROBOTC.
You may choose to use any other LEGO MINDSTORMS Education EV3 compatible text-based programming languages.

LEGO Education has no ownership of the ROBOTC platform and does not provide any support or guarantee of the quality of the user experience and technology used. All required set up information is provided by ROBOTC at http://www.robotc.net/. We recommend always to reinstall the official LEGO MINDSTORMS EV3 Brick firmware when you finish using other programming languages.

POSSIBLE SOLUTION
FILENAME: CODING-04_1.C

```
#pragma config(Sensor, S3, colorSensor, sensorEV3_Color, modeEV3Color_
Color)
#pragma config(Motor, motorB, leftMotor, tmotorEV3_Large, PIDControl,
driveLeft, encoder)
#pragma config(Motor, motorC, rightMotor, tmotorEV3_Large, PIDControl,
driveRight, encoder)

/*
Create a program that drives the robot forward until the Color Sensor
sees red.
The robot then stops.
*/

task main()
{
   //Set motor speed at 20% (Drive Forwards).
   setMotorSpeed(motorB, 20);
   setMotorSpeed(motorC, 20);

   //Loop while the Color Sensor does see red.
   while(getColorName(colorSensor) == colorRed)
      {
       //Keep driving while the Color Sensor does see red.
       sleep(10);
      }

    //Set motor speed to 0% (Stop).
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);
}
```

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

WHAT IS NEXT?

Detecting Objects

Line Detection

Student Worksheets

Design ways to improve driving safety by helping to prevent drivers from falling asleep and causing an accident.

Student Worksheets

CONNECT

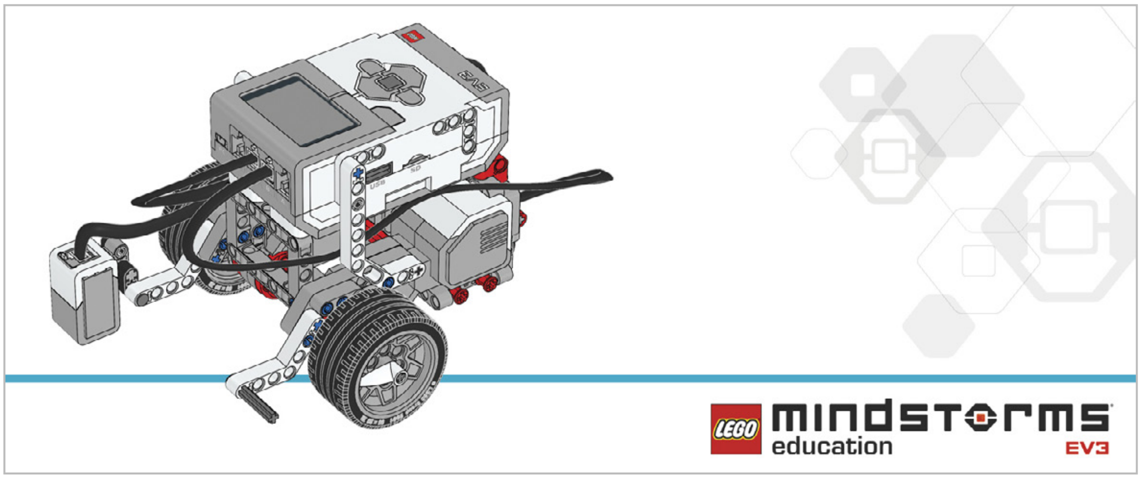
Make sure that you can answer the following questions:
- Can autonomous cars react to different traffic light signals?
- What can happen if a driver falls asleep while driving?
- How can we detect when a driver is falling asleep?

Think about what you have learned, then document it. Describe the problem in your own words. Creatively record your ideas and findings.

CONSTRUCT

Build

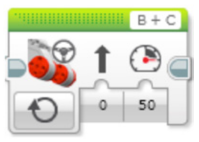
Start by constructing this model.

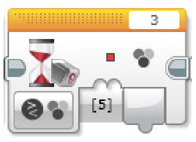


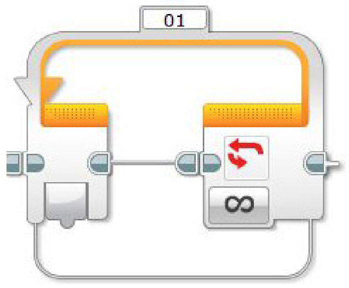
Program

Autonomous cars need to recognize and respond to traffic lights automatically.
Create a program that will make your robot stop at red lights.
Refine your program by making it stop at an appropriate distance from the traffic lights.
Make sure your robot is only responding to the color red.
Once you have succeeded, program your robot to drive forward again when the light changes from red to green.
Use the Color Sensor and the Switch Block to make decisions (Boolean logic). These two blocks will allow the robot to make choices based on the colors it sees.

Consider using these blocks in your solution:







Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas and findings.

CONTEMPLATE

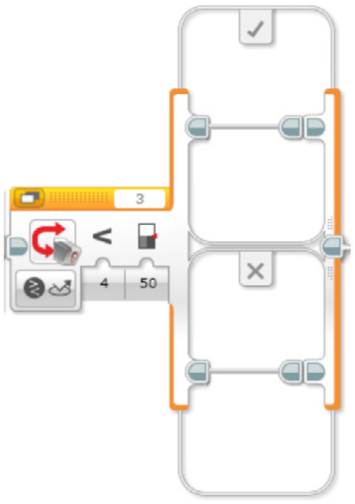
To simulate what could happen if a driver falls asleep while driving, your robot could sound an alarm signal when it crosses the line. This feature is often available in new cars.
Program your robot to perform this function.

Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas, and findings.

Differentiation option

Program your robot to drive on “autopilot” along a given route. You will need to create a program that recognizes and responds to a dark line (or white line). You will create a line-following program and your robot will need to travel along the line without losing contact with it.
You will need to constantly debug your program in order to make your robot travel as smoothly as possible along the line.

Consider adding this blocks to your solution:



Reflect and document your learnings. Describe your pseudo-code for this task. Creatively record your ideas, and findings.

Share

Think about what you have learned, then document it. Creatively record and present your ideas, creations, and findings.

Consider the following questions:
What challenged you? Where there any surprises? How could you improve your program?
Could your program have been more streamlined? Have you used too many blocks?
Is there a more efficient way of building your program?
How could your program be used in real-world scenarios?

CONTINUE

Explore text-based programming solutions for this lesson and compare these solutions using different programming languages.

Object Detection

Design ways to avoid accidents between vehicles and objects in the road.

Learning Objectives

Students will:

Extend understanding of the Loop

Vocabulary

Input, output, algorithm, pseudocode, debug

Grades

6-8

Subjects

Engineering, STEM, Coding

Duration

45-90 Minutes

Difficulty

Intermediate

Standards

NGSS
MS-ETS1-1. / MS-ETS1-2. / MS-ETS1-3. /MS-ETS1-4

CSTA
2-A-2-1 / 2-A-7-2 / 2-A-7-3 / 2-A-7-4 / 2-A-5-5 / 2-A-5-6 / 2-A-3-9 / 2-A-6-10 / 2-C-7-11 / 2-C-4-12 / 2-D-5-16 / 2-I-1-20

Materials Needed

LEGO® MINDSTORMS Education EV3 core set
LEGO MINDSTORMS EV3 Software or Programming app
ROBOTC software (optional)

Object Detection

CONNECT (5 minutes)

Ignite a classroom discussion around the following questions:

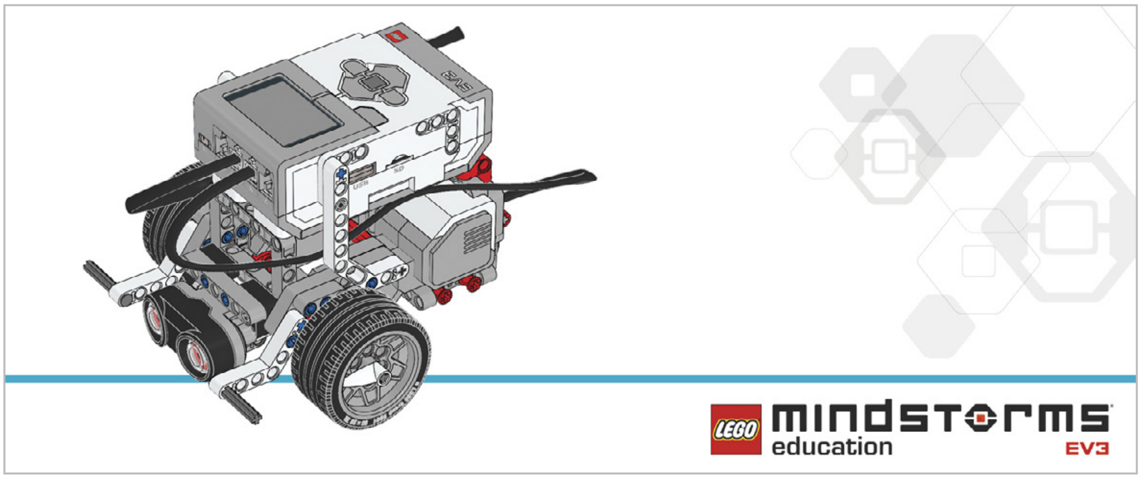
- In what driving situations can a car hit an obstacle?
- What factors are crucial to be aware of in order to avoid collisions with obstacles?
- What causes traffic jams in high density areas?

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their ideas. Encourage them to document their thoughts using text, videos, images, sketchnotes, or another creative medium.

CONSTRUCT (15 to 30 minutes)

Build

Students will construct the Robot Educator base model, then they will add the Ultrasonic Sensor pointing forward.



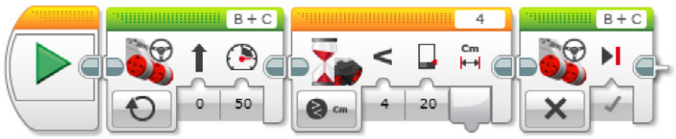
Have the students perform the following building check before they program their robots:

- Are the wires correctly connected from the motors to ports B and C?
- Are the wheels correctly installed?
- Are the wheels rotating freely?
- Are the wires correctly connected from the Ultrasonic Sensor to 4?

Program

Introduce the simple use of the Ultrasonic Sensor.
Demonstrate the Wait Block and how to use it with the Ultrasonic Sensor.
Ask the students how they could make a program to detect any obstacles that might appear while the wheeled robot is moving forward (or backward).
Students will create a program that makes the robot stop at a given point based on a distance measured by the Ultrasonic Sensor.
Allow the students to select the tool(s) they find most appropriate for capturing and sharing their pseudocode. Encourage them to use text, videos, images, sketchnotes, or another creative medium.

POSSIBLE SOLUTION
FILENAME: CODING-05.EV3 (Tab:1)



DETECT OBJECTS AND STOP
1. Start the program.
2. Turn both motors on at speed 50.
3. Wait for the Ultrasonic Sensor to detect an obstacle at a distance of less than 20 cm.
4. Turn both motors off.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

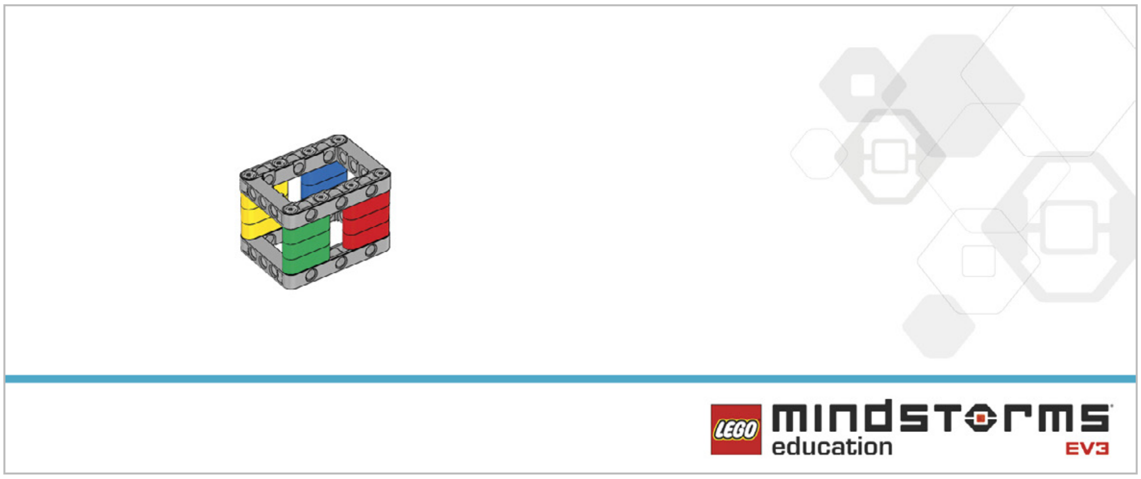
Note

Refer students to the Robot Educator Tutorials for further assistance.

In the EV3 Software :
Robot Educator > Basics > Straight Move
Robot Educator > Basics > Stop at Object

Setup

Before assigning the next task, make sure that you have marked the path the robots must follow and that there is enough space to complete the task. It is a good idea to have students work on a big table or on the floor.
Have the students experiment with using the Ultrasonic Sensor to detect different objects. One of the objects can be a cuboid made of LEGO® bricks.

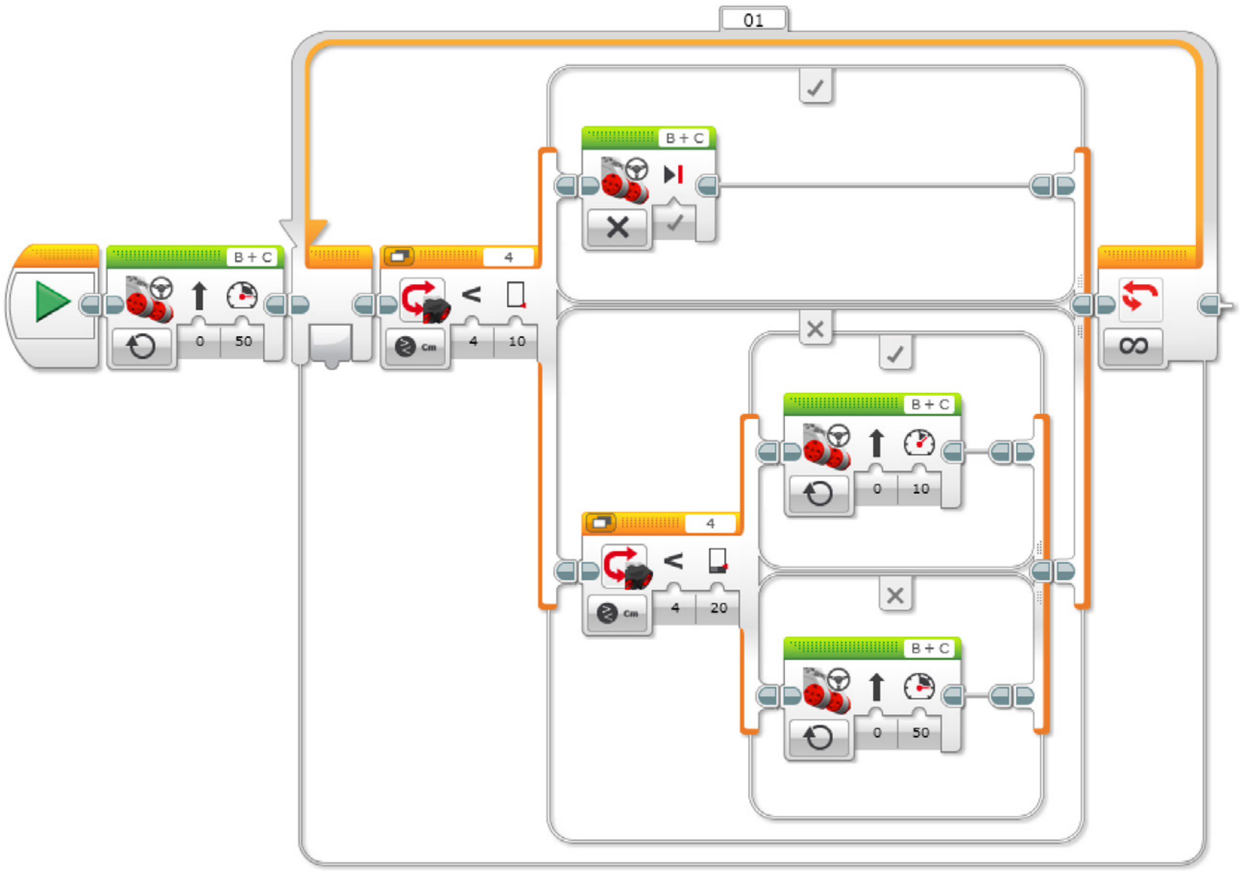


CONTEMPLATE (35 minutes)

On the road, when a driver sees an object, they slow their car down before coming to a full stop.
Have the students program their robots with the same behavior.
If the Ultrasonic Sensor:

- Detects an object less than 10 cm away, make the robot stop
- Detects an object between 10 and 20 cm away, make the robot slow down
- Does not detect any object, continue to move at full speed

POSSIBLE SOLUTION
FILENAME: CODING-05.EV3 (Tab:2)



DETECT OBJECTS AND REACT
1. Start the program.
2. Turn both motors on at speed 50.
3. IF the Ultrasonic Sensor detects an obstacle at a distance of less than 10 cm, turn both motors off.
ELSE
4. IF the Ultrasonic Sensor detects an obstacle at a distance of less than 20 cm, turn both motors on at speed 10.
ELSE
5. Turn both motors on at speed 50.
6. Repeat steps 3 to 7 forever.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Differentiation Option

Bring all of the teams together.
Tell the students to place their robots in a vertical line with varying amounts of space between them (just like cars in a traffic jam).
Have them start their programs at the same time and watch what happens.
Ask the students to refine their programs so that all of the robots continue driving at the same speed with equal distances between them (like well-flowing traffic on a road).

Share

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their creations, unique thinking, and learning process. Encourage them to use text, videos, images, sketchnotes, or another creative medium.
Discuss the concept of efficiency in programming.
How many variations did the group as a whole come up with? Compare the many possible solutions to the given problem.

Assessment Opportunity
Specific rubrics for assessing computational think can be found under ‘Assessment’.

CONTINUE (45 minutes)

Using text-based programming

Have the students explore text-based programming solutions so they can compare different programming languages.

IMPORTANT
The following is a possible solution using the text-based programming language ROBOTC.
You may choose to use any other LEGO MINDSTORMS Education EV3 compatible text-based programming languages.

LEGO Education has no ownership of the ROBOTC platform and does not provide any support or guarantee of the quality of the user experience and technology used. All required set up information is provided by ROBOTC at http://www.robotc.net/. We recommend always to reinstall the official LEGO MINDSTORMS EV3 Brick firmware when you finish using other programming languages.

POSSIBLE SOLUTION
FILENAME: CODING-01_1.C

```
#pragma config(Sensor, S4,      sonarSensor,     sensorEV3 _ Ultrasonic)
#pragma config(Motor,  motorB, leftMotor,      tmotorEV3 _ Large,
PIDControl, driveLeft, encoder)
#pragma config(Motor,  motorC, rightMotor,     tmotorEV3 _ Large,
PIDControl, driveRight, encoder)

/*
Create a program that drives the robot forward until the Ultrasound
Sensor sees an object.
The robot then stops.
*/

task main()
{
  //Set motor speed at 50% (Drive Forwards).
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);
  while(getUSDistance(sonarSensor) < 20)
  {
        //Keep going until the Ultrasonic Sensor sees a value less than
20cm.
  }

  //Once the Ultrasonic Sensor sees a value less than 20cm.
  //Set motor speed to 0% (Stop).
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
}
```

Program solutions for this lesson are available for download at:
http://www.legoeducation.com

WHAT IS NEXT?

Unlocking a Car

Object Detection

Student Worksheets

Design ways to avoid accidents between vehicles and objects in the road.

Student Worksheets

CONNECT

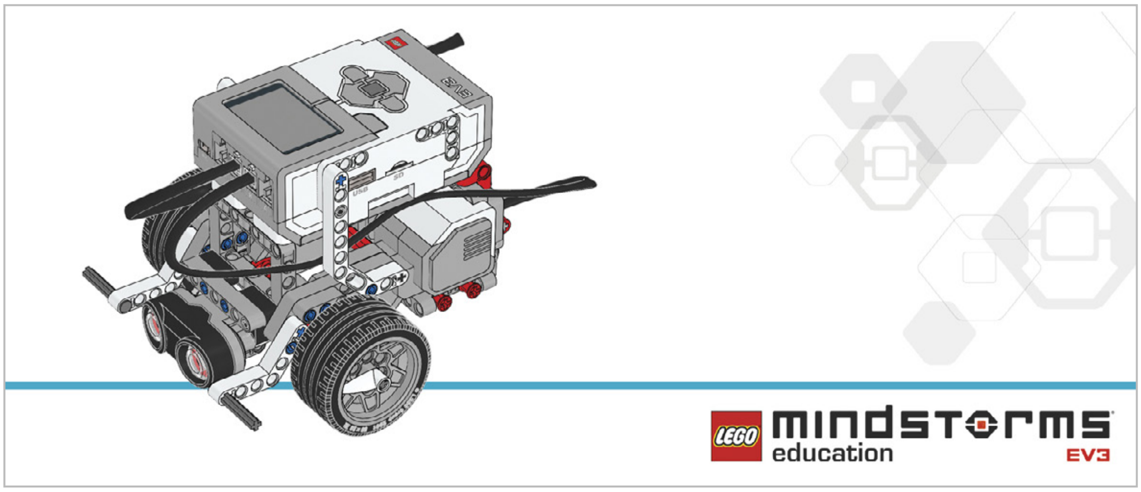
Make sure that you can answer the following questions:
- In what driving situations can a car hit an obstacle?
- What factors are crucial to be aware of in order to avoid collisions with obstacles?
- What causes traffic jams in high density areas?

Think about what you have learned, then document it. Describe the problem in your own words. Creatively record your ideas and findings.

CONSTRUCT

Build

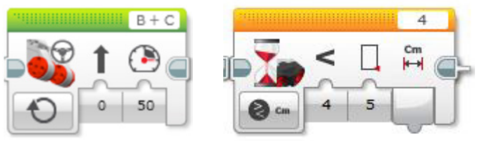
Start by constructing this model.



Program

Program your robot to detect any obstacles that might appear while the robot is moving forward (or backward).
Make the robot stop when it detects an object that is less than 20 cm away.

Consider using these blocks in your solution:



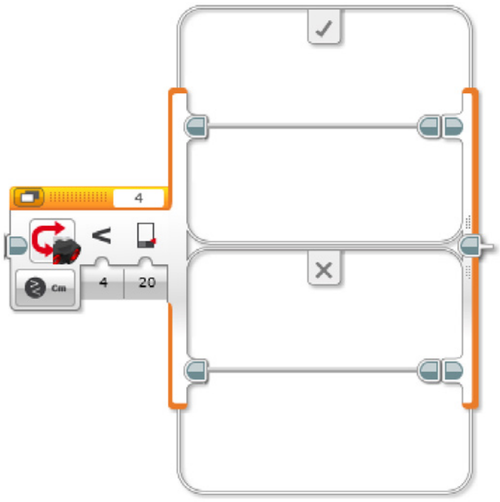
Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas and findings.

CONTEMPLATE

On the road, when a driver sees an object, they slow their car down before coming to a full stop.
Program your wheeled robot to do the same.
If the Ultrasonic Sensor:.

- Detects an object less than 10 cm away, make the robot stop
- Detect an object between 10 and 20 cm away, make the robot slow down
- Does not detect any object, continue to move at full speed

Consider adding these blocks to your solution:



Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas, and findings.

Differentiation option

Get together with the other teams.
Place all of the robots in a vertical line with varying amounts of space between them (just like cars in a traffic jam).
Have everyone start their programs at the same time and see what happens.
Refine your program so that all of the robots continue driving at the same speed with equal distances between them (like well-flowing traffic on a road).

Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas, and findings.

Share
Consider the following questions:
What does “efficiency in programming” mean?
How many variations did the class as a whole come up with? Compare the many possible solutions to the given problem.

CONTINUE
Explore text-based programming solutions for this lesson and compare these solutions using different programming languages.

Unlocking a Car

Design a way to use passcodes to protect cars from thieves.

Learning Objectives

Students will:

Understand simple Boolean logic (such as AND, OR and NOT) and some of its uses in circuits and programming
Use several inputs in combination.

Vocabulary

Input, output, pseudocode, Boolean logic, algebraic, switch, true, false, loop

Grades

6-8

Subjects

Engineering, STEM, Coding

Duration

45-90 Minutes

Difficulty

Advanced

Standards

NGSS
MS-ETS1-1. / MS-ETS1-2. / MS-ETS1-3. /MS-ETS1-4

CSTA
2-A-2-1 / 2-A-7-2 / 2-A-7-3 / 2-A-7-4 / 2-A-5-5 / 2-A-5-6 / 2-A-5-7 / 2-A-3-9 / 2-A-6-10 / 2-C-7-11 / 2-C-4-12 / 2-D-5-16 / 2-I-1-20

Materials Needed

LEGO® MINDSTORMS Education EV3 core set
LEGO MINDSTORMS EV3 Software or Programming app
ROBOTC software (optional)

Unlocking a Car

CONNECT (5 minutes)

Did you know that it's now possible to start a car without a key? The "car key" or fob allows the engine to be started remotely. The driver just needs to press a combination of buttons and the brake or clutch in order to start the car. In this lesson, students will learn how they can do the same using a combination of sensors and algebraic logic to start their wheeled robots.
Ignite a classroom discussion around the following questions:

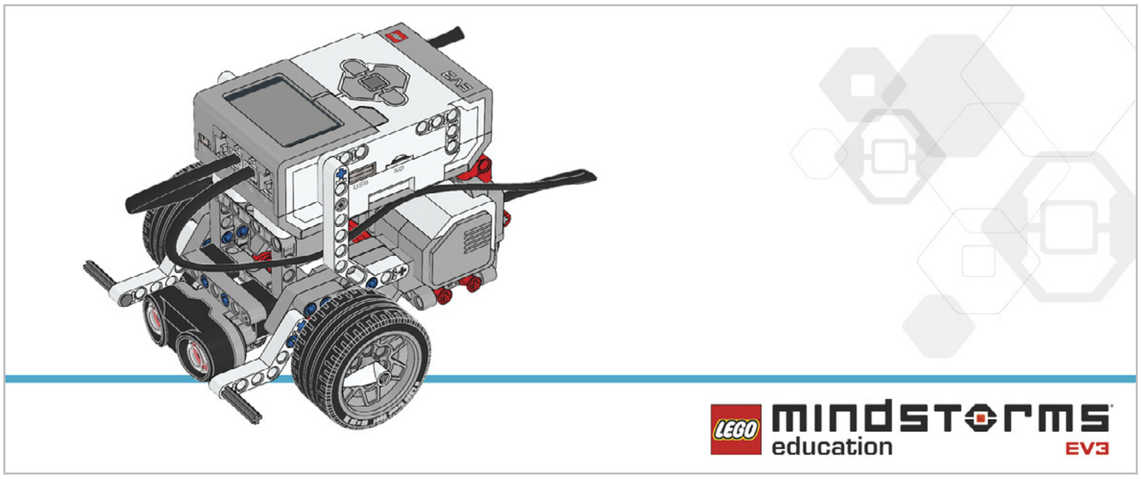
- How do electronic keys work?
- How can we ensure that each key is only able to open one car?

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their ideas. Encourage them to document their thoughts using text, videos, images, sketchnotes, or another creative medium.

CONSTRUCT (15 to 30 minutes)

Build

Students will construct the Robot Educator base model, then they will add the Ultrasonic Sensor.



Have the students perform the following building check before they program their robots.

- Are the wires correctly connected from the motors to ports B and C?
- Are the wheels correctly installed?
- Are the wheels rotating freely?
- Are the wires correctly connected from the Ultrasonic Sensor to 4?

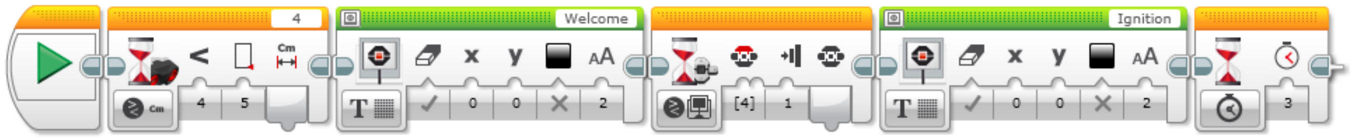
Program

Have the students begin a new project in the EV3 programming environment.
Have them program their robot to first detect the presence of the driver and then that the driver has pressed the ignition button. In this example:

- The Ultrasonic Sensor is used to simulate the key in the driver's pocket when the driver is in the car. This is known as "entering the bubble".
- The Brick Button is used to turn on the car.
- The Brick Display is used as the dashboard to display messages.

Once the students have gotten their wheeled robot to display messages, ask them to place their program inside a loop in order to repeat those messages.
The suggested program uses two inputs in a sequence.
Allow the students to select the tool(s) they find most appropriate for capturing and sharing their pseudocode. Encourage them to use text, videos, images, sketchnotes, or another creative medium.

POSSIBLE SOLUTION
FILENAME: CODING-06.EV3 (Tab:1)



PRESENCE DETECTION AND CAR IGNITION IN SEQUENCE
1. Start the program.
2. Wait for the Ultrasonic Sensor to detect an object at a distance of less than 5 cm.
3. Show a welcome message on the Brick Display.
4. Wait for the Up Button to be pressed.
5. Show an ignition message on the Brick Display.
6. Wait 3 seconds, then end the program.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Note

Refer students to the Robot Educator Tutorials for further assistance.

In the EV3 Software :

Robot Educator > Beyond Basics > Logic
Robot Educator > Basics > Stop at Object
Robot Educator > Beyond Basics > Loop
Robot Educator > Beyond Basics > Switch
Robot Educator > Beyond Basics > Data Wires
Robot Educator > Beyond Basics > Sensor Blocks

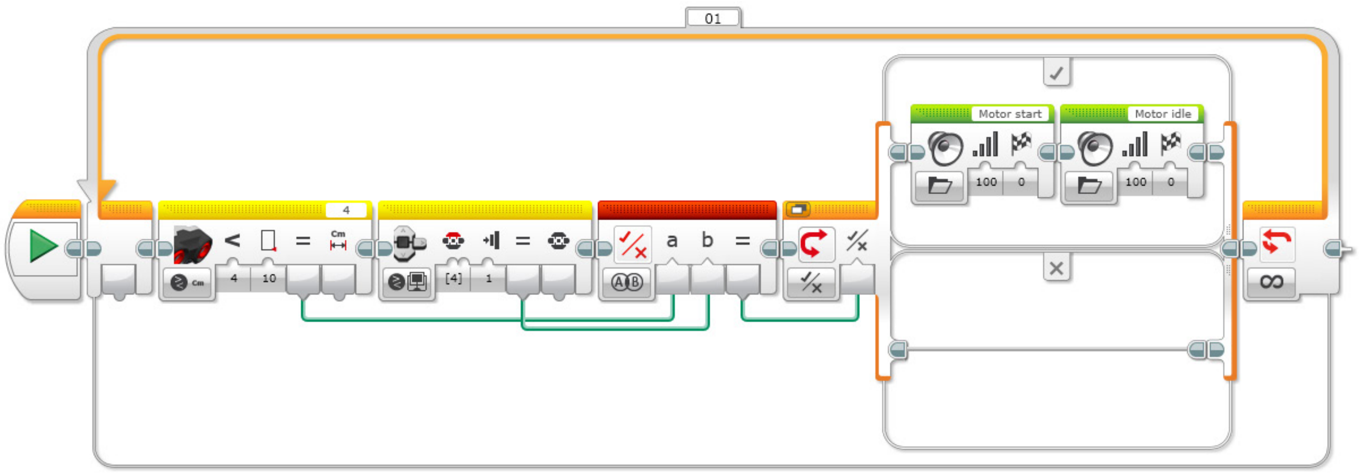
CONTEMPLATE (35 minutes)

Students will now have to use the two inputs simultaneously.
Have the students create a program that only grants control of the robot once the right start combination is activated.
Have the students reflect on their first program. Ask them how they can create a code to activate the robot. Guide them through the use of the Logic Operations Block. Explain that the Logic Operations Block is similar to a Venn diagram in the way it behaves. Point out the many conditions this diagram has and how it relates to the Logic Operations Block in the EV3 Software. To do this, you might explain the following image and how the results are created.

Modes	Inputs Used	Result
AND	A, B	True if both A and B (or both) are True, False if both A and B are False
OR	A, B	True if both A and B (or both) is True, False if both A and B are False
XOR	A, B	True if one of A or B is True, False if both A and B are True, False if both A and B are False
NOT	A	True if A is False, False if A is True

The EV3 Software Logic Operations Block creates a True/False output depending on the criteria set in the block. For example, A and B gives a True output only when A and B are both True.

POSSIBLE SOLUTION
FILENAME: CODING-06.EV3 (Tab:2)



2 SIMULTANEOUS CONDITIONS TO START A CAR
1. Start the program.
2. The Ultrasonic Sensor sends a True result when it detects an object at a distance of less than 4 cm.
3. The Brick Button sends a True result when the Up Button has been pressed.
4. The Logic Operations Block waits for two True messages to be received before sending a True message to the logic switch.
5. The logic switch compares the message received from the Logic Operations Block.
   If it is True, the EV3 Brick plays two sounds.
   If it is False, nothing happens.
6. Repeat steps 2 to 5 forever.

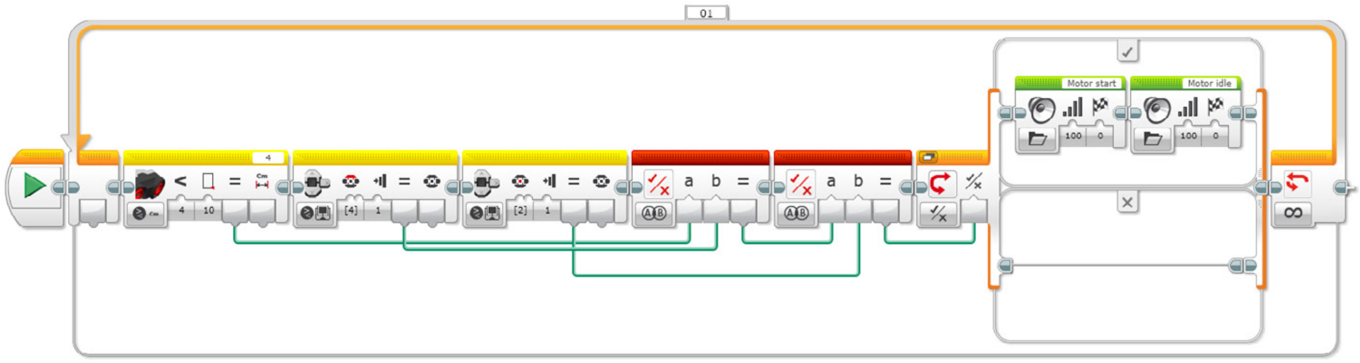
Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Differentiation Option

Have the students design a program to simulate a passcode with three criteria. Make sure the students understand that all three of the criteria must be true in order for the car to start. For example:
   The key or the fob itself needs to be inside the car.
   The brake or clutch needs to be pressed.
   The button must be pressed in order to turn on the ignition.

Students could connect the Touch Sensor to simulate the clutch or brake.
Have the students design a program that displays an error message if all three of the criteria for starting the car aren't met.

POSSIBLE SOLUTION
FILENAME: CODING-06.EV3 (Tab:3)



THREE SIMULTANEOUS CONDITIONS TO START A CAR
1. Start the program.
2. The Ultrasonic Sensor sends a True result when it detects an object at a distance of less than 4 cm.
3. Brick Button sends a True result when the Top button has been pressed.
4. Brick Button sends a True result when the enter button has been pressed.
5. The Logic Operations Block waits for two True messages to be received before sending a True message to the logic switch (messages come from the Ultrasonic Sensor and Touch Sensor).
6. The Logic Operations Block waits for two True messages to be received before sending a True message to the logic switch (messages come from the first Logic Operations Block and the Brick Button).
7. The logic switch compares the messages received from the Logic Operations Block.
   If it is True, the EV3 Brick plays two sounds.
   If it is False, nothing happens.
6. Repeat steps 2 to 7 forever.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Share

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their creations, unique thinking, and learning process. Encourage them to use text, videos, images, sketchnotes, or another creative medium.
How did the students find using the Data Wires? Ask them to share their thoughts. Point out that wires can be moved around and tidied up, which is important to ensure programs don't become too messy.
Have a discussion about the Wait Block and logical operations. Explain that the Wait Block is generally used with one sensor at a time and that the Logic Operations Block allows users to use multiple sensors simultaneously.

Assessment Opportunity

Specific rubrics for assessing computational think can be found under 'Assessment'.

CONTINUE (45 minutes)

Using text-based programming

Have the students explore text-based programming solutions so they can compare different programming languages.

IMPORTANT
The following is a possible solution using the text-based programming language ROBOTC. You may choose to use any other LEGO MINDSTORMS Education EV3 compatible text-based programming languages.

LEGO Education has no ownership of the ROBOTC platform and does not provide any support or guarantee of the quality of the user experience and technology used. All required set up information is provided by ROBOTC at http://www.robotc.net/. We recommend always to reinstall the official LEGO MINDSTORMS EV3 Brick firmware when you finish using other programming languages.

POSSIBLE SOLUTION
FILENAME: CODING-06_1.C

```
#pragma config(StandardModel, "EV3 _ REMBOT")
/*
Create a program that shows how a keyless ignition works. The
Ultrasonic Sensor welcomes the driver. The Touch Sensor works the
ignition.
*/

task touchTask()
{
  //Wait for the touch sensor to be pressed.
  while(getTouchValue(touchSensor) == 0)
  {
       //Do Nothing.
       sleep(10);
  }
  //Display Text on the LCD Screen.
  eraseDisplay();
  displayCenteredBigTextLine(4, "Ignition");
  //Display text for 3 seconds.
  sleep(3000);
}
```

```
task main()
{
  //Start the second task to monitor the touch sensor.
  startTask(touchTask);
  //Wait for the sonar sensor to see an object that is less than or
equal to 5cm away.
  while(getUSDistance(sonarSensor) >= 5)
  {
        //Do Nothing.
        sleep(10);
  }
  //Display text on the LCD Screen.
  eraseDisplay();
  displayCenteredBigTextLine(4, “Welcome”);
  //Display text for 3 seconds.
  sleep(3000);
}
```

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

WHAT IS NEXT?

Cruise Control

Unlocking a car

Student Worksheets

Design a way to use passcodes to protect cars from thieves.

Student Worksheets

CONNECT

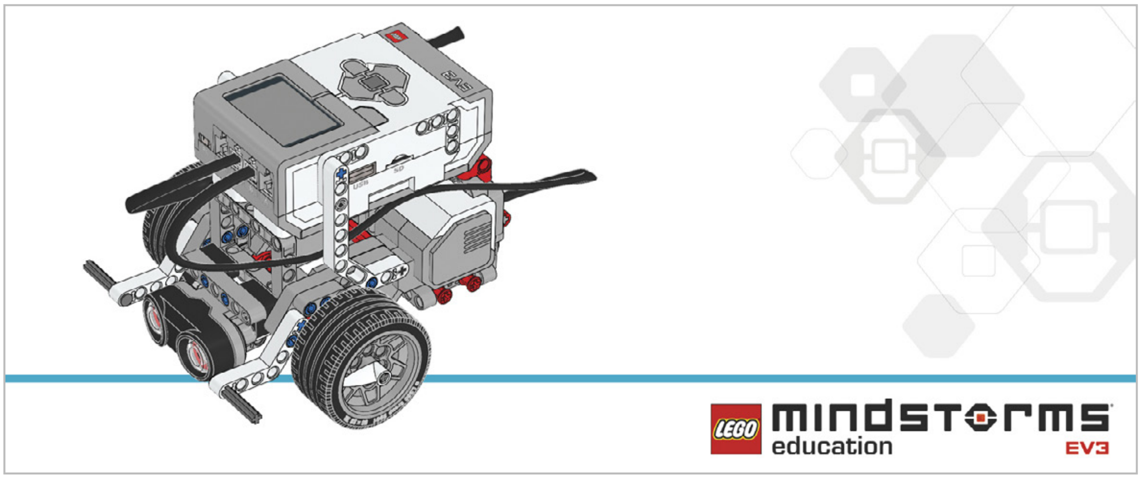
Make sure that you can answer the following questions:
- How do electronic keys work?
- How can we ensure that each key is only able to open one car ?

Think about what you have learned, then document it. Describe the problem in your own words. Creatively record your ideas and findings.

CONSTRUCT

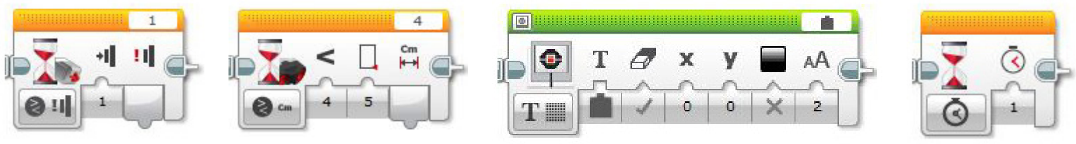
Build

Start by constructing this model.



Program

Create a keyless entry system for your robot.
When a combination of sensors is activated, your drive program will be executed.
Program your robot to display the text “Welcome” when the Ultrasonic Sensor detects an object, and then display the text “Ignition” when the Touch Sensor is pressed.
Consider using these blocks in your solution:



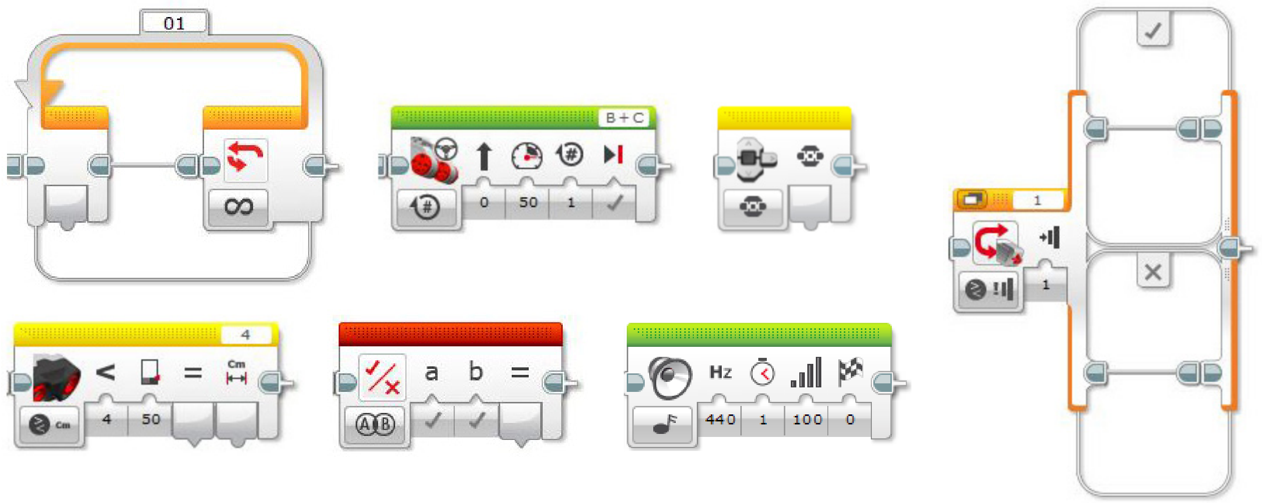
Reflect and document your learnings. Describe your pseudo-code for this task. Creatively record your ideas, and findings.

CONTEMPLATE

You will need to use a number of different inputs and logic operations to make sure that the two inputs work together to provide information to another block. Think about how a keyless car works. For this task:

- The Touch Sensor is the ignition.
- The Ultrasonic Sensor is used to detect the key inside the car.
- Both sensors need to be activated correctly in order for your robot to start.
- Use the sound output of your robot to indicate that it has started.

Consider adding these blocks to your solution:



For this task, you will use the sensor blocks (yellow) to create logic for the Logic Operations Block. Each sensor block will be used to create a True/False output.

Reflect and document your learnings. Describe your pseudo-code for this task. Creatively record your ideas, and findings.

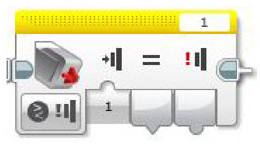
Differentiation option

You now need to program your robot to "start" only when the conditions of three different sensors have all been met. For this task:

- The Touch Sensor is the ignition.
- The Ultrasonic Sensor is used to detect the key inside car.
- The Brick Buttons are used as the brake/clutch.

Each Logic Operations Block can receive two inputs. But what happens if we want three inputs? Think about using two Logic Operations Blocks to achieve this. Two sensors will need to enter the first Logic Operations Block. The output is then taken to the next Logic Operations Block with the third input (sensor). That result is then taken to the Switch Block.

Consider adding these blocks to your solution:



Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas and findings.

Share

Consider the following questions:
What did you think of the experience of using many Data Wires?
Could your program have been more streamlined? Have you used too many blocks?
Is there a more efficient way of building your program?
How could your program be used in real-world scenarios?
Comparing text-based programming with visual programming, which is easier to follow?
If you have not tried both programming methods, try the other method to see which is more efficient.

CONTINUE

Explore text-based programming solutions for this lesson and compare these solutions using different programming languages.

Cruise Control

Design a cruise control program to assist drivers by making their driving experience less stressful.

Learning Objectives

Students will:

Use the Variable Block to store information
Develop multi-level programs
Create function blocks (My Blocks)

Vocabulary

Input, output, pseudocode, variable, constant, loop, wait, motor, Touch Sensor

Grades

6-8

Subjects

Engineering, STEM, Coding

Duration

45-90 Minutes

Difficulty

Advanced

Standards

NGSS
MS-ETS1-1. / MS-ETS1-2. / MS-ETS1-3. /MS-ETS1-4

CSTA
2-A-2-1 / 2-A-7-2 / 2-A-7-3 / 2-A-7-4 / 2-A-5-5 / 2-A-5-6 / 2-A-5-7 / 2-A-3-9 / 2-A-6-10 / 2-C-7-11 / 2-C-4-12 / 2-D-5-16 / 2-I-1-20

Materials Needed

LEGO® MINDSTORMS Education EV3 core set
LEGO MINDSTORMS EV3 Software or Programming app
ROBOTC software (optional)

Cruise Control

CONNECT (5 minutes)

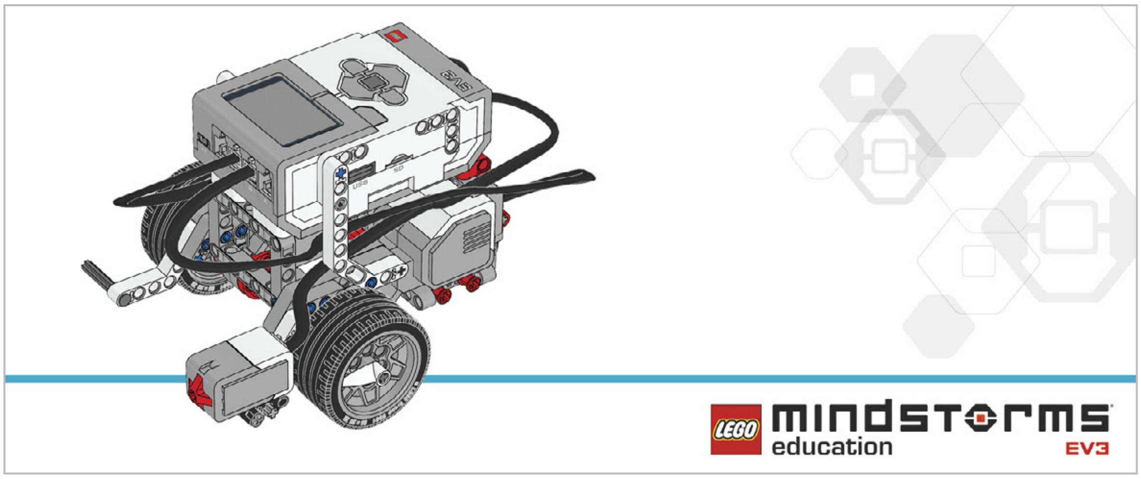
During this lesson, you will introduce the students to variables. The students will use this information to create a cruise control feature for their wheeled robots. Pressing the Touch Sensor will increase the speed of the robot.
Ignite a classroom discussion around the following questions:
- What factors can make drivers feel stressed while driving?
- How can we help improve drivers' safety during long drives?

Ask the students what they think the plus and minus buttons mean on a steering wheel. Discuss how these can adjust the speed of the car.
Allow the students to select the tool(s) they find most appropriate for capturing and sharing their ideas. Encourage them to document their thoughts using text, videos, images, sketchnotes, or another creative medium.

CONSTRUCT (15 to 30 minutes)

Build

Students will construct the Robot Educator base model, then they will add **two** Touch Sensors.



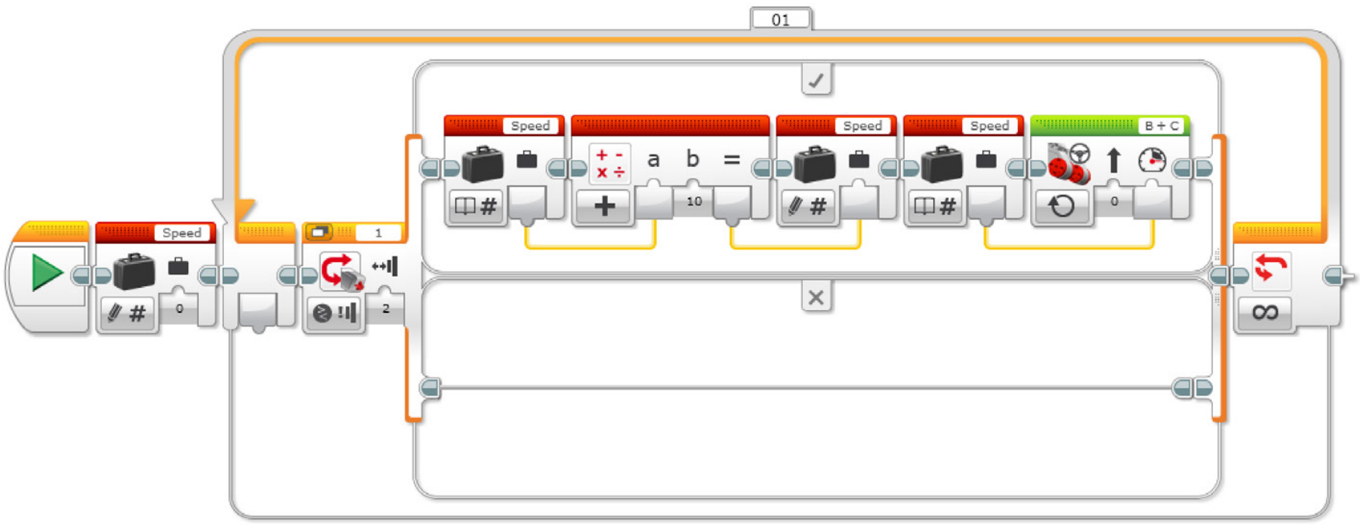
Tell the students that today's task will utilize two Touch Sensors to control and maintain the speed of their robot.

Have the students perform the following building check before they program their robots:
- Are the wires correctly connected from the motors to ports B and C?
- Are the wheels correctly installed?
- Are the wheels rotating freely?

Program

Have the students begin a new project in the EV3 programming environment.
Expand on the students' understanding of the Variable Block. Explain that it is a programming block that can store data (text, logic, a numeric, or arrays), which can be overwritten at any time while the program is running.
Explain that this block has to be read and then written to, using the Math, Text, or Array Operations Blocks.
The students will need to program their robot so that, once moving, it can be sped up by pressing the Touch Sensor. Code blocks for controlling the Touch Sensor (wait) should be placed inside a loop.
Allow the students to select the tool(s) they find appropriate for capturing and sharing their pseudocode. Encourage them to use text, videos, images, sketchnotes, or another creative medium.

POSSIBLE SOLUTION
FILENAME: CODING-07.EV3 (Tab:1)



INCREACE SPEED WITH VARIABLE

```
1. Start the program.
2. Create a Variable Block called "Speed" and enter a value of 0.
3. If the Touch Sensor is pressed:
   a. Read the variable called "Speed"
   b. Add 10 to the read value
   c. Write the result in the variable called "Speed"
   d. Read the variable called "Speed"
   e. Start motors B and C at a speed set to the value stored in the variable called "Speed"
ELSE
   (Do nothing)
4. Repeat steps 3a to 3e forever.
```

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Note

Refer students to the Robot Educator Tutorials for further assistance.

In the EV3 Software :

Robot Educator > Beyond Basics > Variables
Robot Educator > Tools > My Blocks
Robot Educator > Basics > Straight Move
Robot Educator > Beyond Basics > Multitasking
Robot Educator > Beyond Basics > Loop
Robot Educator > Beyond Basics > Data Wires
Robot Educator > Beyond Basics > Math - Basic

Explain that the students will be using the Variable Block today.
You might wish to explain to the students the difference between a constant and a variable.

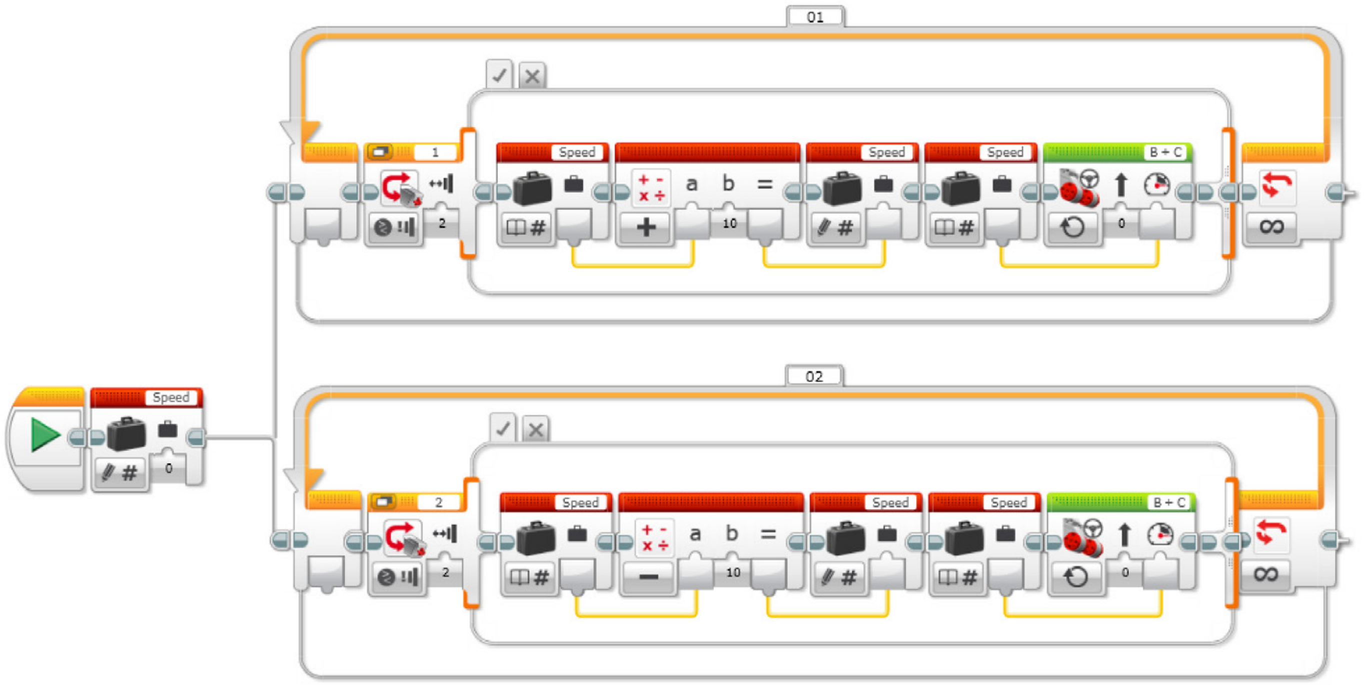
- A constant is used to provide values in a program over and over again. These fixed values can only be edited by the user when the program is not running.
- A variable is a way of storing values in a program that can be used in that program. The difference here is that the value can be overwritten time and time again as the program is running.

You may want to show the Constant Block in action with the Move Steering Block and Display Block.
Show students how to use the Variable Block, and get them to create a program that contains a variable.

CONTEMPLATE (35 minutes)

With the first program written and the robot accelerating when the Touch Sensor is pressed, ask the students to think about how they would extend the program to slow the robot down.
One solution could be to have a second unlimited loop similar to the loop used in the first programming exercise. This loop would use a different Touch Sensor port (another sensor added).
The Math Block would be changed to subtract rather than add.
Remind the students about multitasking and tell them that they will need to use their knowledge of this type of programming in this lesson.

POSSIBLE SOLUTION
FILENAME: CODING-06.EV3 (Tab:2)



INCREASE AND DECREASE SPEED WITH VARIABLE

1. Start the program.
2. Create a Variable Block called “Speed”, enter a value of 0, and start two tasks.

TASK 1

3. If Touch Sensor 1 is pressed:
   a. Read the variable called “Speed”
   b. Add 10 to the read value
   c. Write the result in the variable called “Speed”
   d. Read the variable called “Speed”
   e. Start motors B and C at a speed set to the value stored in the variable called “Speed”

ELSE
   (Do nothing)

4. Repeat steps 3a to 3e forever.

TASK 2

5. If Touch Sensor 2 is pressed:
   a. Read the variable called “Speed”
   b. Subtract 10 from the read value
   c. Write the result in the variable called “Speed”
   d. Read the variable called “Speed”
   e. Start motors B and C at a speed set to the value stored in the variable called “Speed”

ELSE
   (Do nothing)

6. Repeat steps 5a to 5e forever.

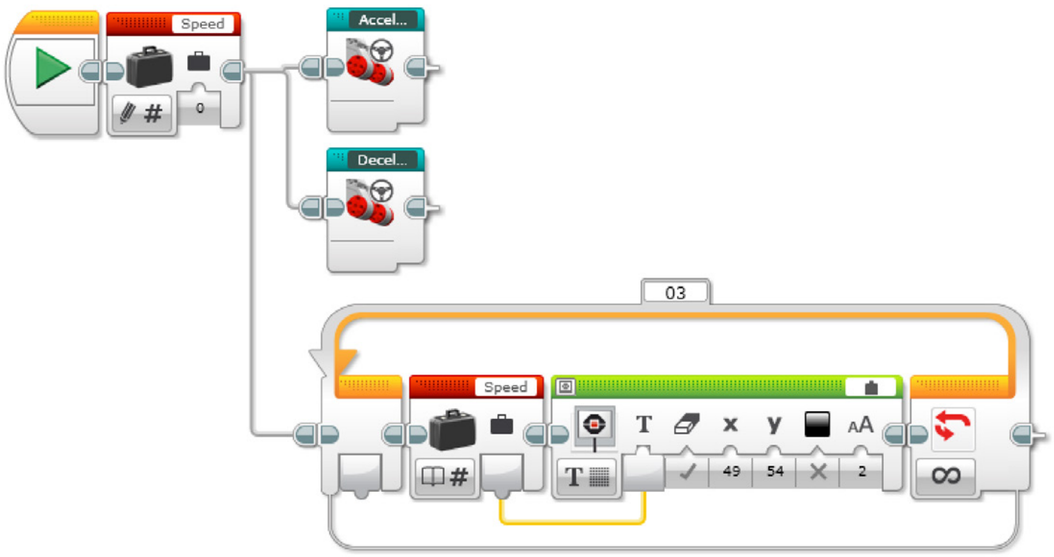
Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Differentiation Option

With the wheeled robot now accelerating and decelerating at the touch of a button (or two), students can extend their programming to show how fast their robots are moving. Students will use the Display Block to show this value in the Variable Block.

A new skill to be learned in this lesson is the creation of a My Block. Two of these can be seen in the solution below. My Blocks allow users to create subroutines of programs they have already written. In the case below, we have taken the acceleration and deceleration loops and created My Blocks from these programs. There are two reasons to do this: first to save space, and second to allow these subroutines to be reused in other programs. A tutorial on My Blocks can be found in the Robot Educator section of the EV3 Software.

POSSIBLE SOLUTION
FILENAME: CODING-06.EV3 (Tab:3)



INCREASE AND DECREASE SPEED WITH VARIABLE AND DISPLAY

1. Start the program.
2. Create a Variable Block called “Speed”, enter a value of 0, and start three tasks.

TASK 1

3. Start My Block “Acceleration”.

TASK 2

4. Start My Block “Deceleration”.

TASK 3

5. Read the variable called “Speed”.
6. Display the value stored in the variable called “Speed”.
7. Repeat steps 5 and 6 forever.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Share
Allow the students to select the tool(s) they find most appropriate for capturing and sharing their creations, unique thinking, and learning process. Encourage them to use text, videos, images, sketchnotes, or another creative medium.

Assessment Opportunity
Specific rubrics for assessing computational think can be found under ‘Assessment’.

CONTINUE (45 minutes)
Using text-based programming

Have the students explore text-based programming solutions so they can compare different programming languages.

IMPORTANT
The following is a possible solution using the text-based programming language ROBOTC. You may choose to use any other LEGO MINDSTORMS Education EV3 compatible text-based programming languages.

LEGO Education has no ownership of the ROBOTC platform and does not provide any support or guarantee of the quality of the user experience and technology used. All required set up information is provided by ROBOTC at http://www.robotc.net/. We recommend always to reinstall the official LEGO MINDSTORMS EV3 Brick firmware when you finish using other programming languages.

POSSIBLE SOLUTION
FILENAME: CODING-07_1.C

```
#pragma config(Sensor, S1,     touchSensor,     sensorEV3 _ Touch)
#pragma config(Sensor, S2,     touchSensor2,    sensorEV3 _ Touch)
#pragma config(Motor,  motorB,          rightMotor,    tmotorEV3 _ Large,
PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC,          leftMotor,     tmotorEV3 _ Large,
PIDControl, driveLeft, encoder)

/*
Create a program to control the positive speed of the robot by a press
of a Touch Sensor.
*/

task main()
{
  //Create an integer (whole number) variable to store speed value.
  int speed = 0;
  //Repeat our control loop forever.
  while(true)
  {
      //When I press the touch sensor button.
      if(getTouchValue(touchSensor) == 1)
      {
      //Add 10 to our 'speed' variable
          if(speed < 100) speed = speed + 10;
          //Create a loop to wait for the touch sensor button to be
released.
          while(getTouchValue(touchSensor) == 1)
          {
          //Wait for button to be released.
          sleep(10);
          }

          //Set motorB and motorC speed to the value of the 'speed'
variable
      setMotorSpeed(motorB, speed);
      setMotorSpeed(motorC, speed);
      }
  }
}
```

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

WHAT IS NEXT?

Roaming Vehicles

Cruise Control

Student Worksheets

Design a cruise control program to assist drivers by making their driving experience less stressful.

Student Worksheets

CONNECT

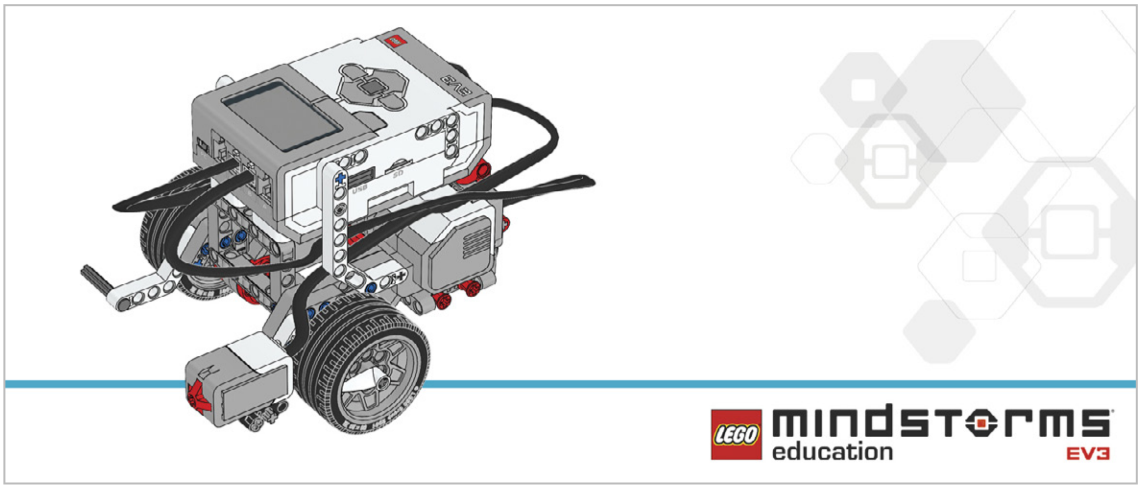
Make sure that you can answer the following questions:
- What factors can make drivers feel stressed while driving?
- How can we help improve drivers' safety during long drives?

Think about what you have learned, then document it. Describe the problem in your own words. Creatively record your ideas and findings.

CONSTRUCT

Build

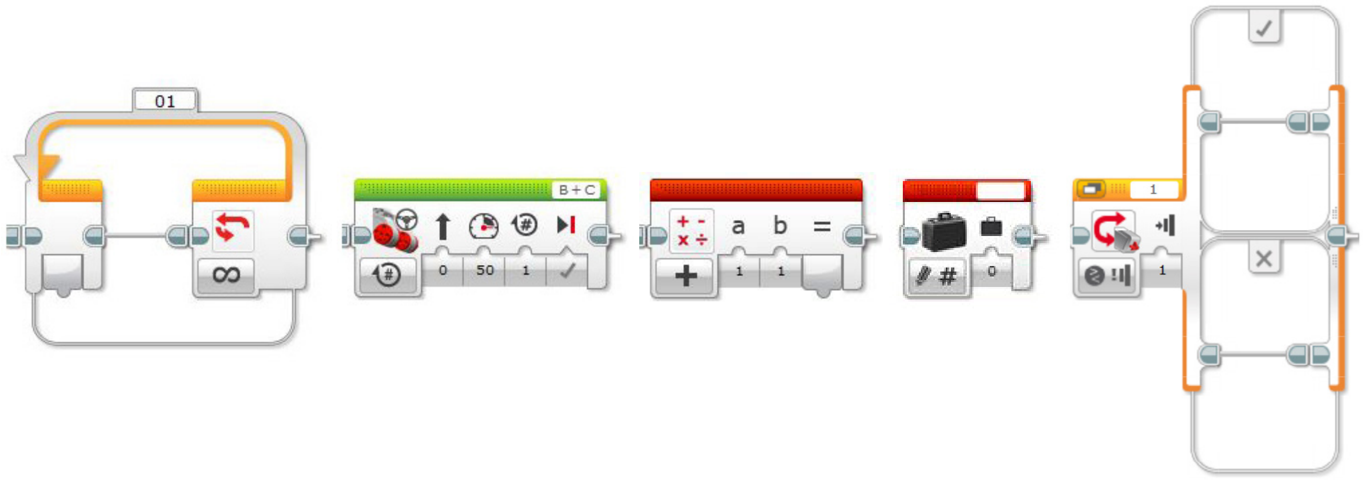
Start by constructing this model.



Program

Create a cruise control program for your robot, like the ones found in many cars today. You will need to use two Touch Sensors to simulate the buttons found on the steering wheel of a car with cruise control.
The car will speed up in increments of ten when the Touch Sensor is pressed.
Use the Variable Block to define the "set" speed that can be added to. Ensure that the Move Steering Block mode is set to On instead of On for Seconds, Degrees, or Rotations.

Consider using these blocks in your solution:

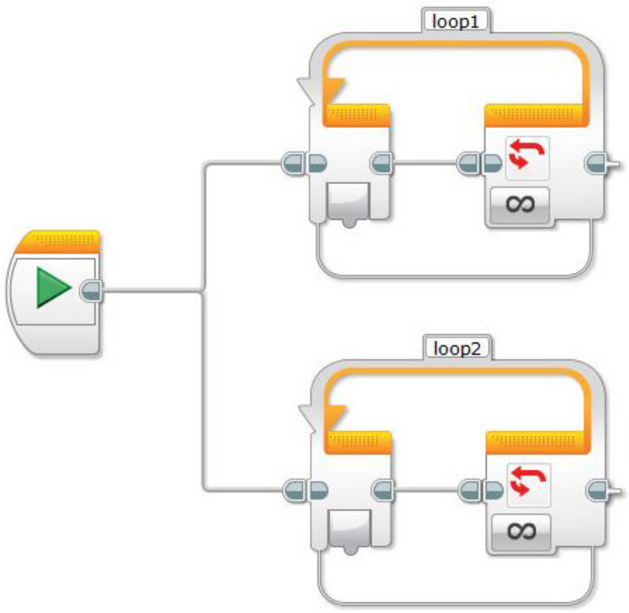


Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas and findings.

CONTEMPLATE

Improve your program: make your robot decelerate in increments of ten when the other Touch Sensor is pressed.
This can be done by adding a second loop and a Switch Block.
Inside the extra loop will be a second Touch Sensor, and a Math Block set to subtract rather than add. Remember that you will be using multitasking with two lines of programming running simultaneously.

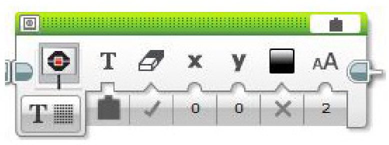
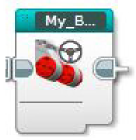
Consider adding these blocks to your solution:



Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas, and findings.

Differentiation option

Improve your program: show the robot's speed (motor power) on the EV3 Brick Display.
Consider adding these blocks to your solution:

My Blocks might be useful at this point because they can save room on the programming screen, and subroutines (the My Blocks) can be used again in other programs you write. Ask your teacher for help if you would like to learn how to use My Blocks.

Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas and findings.

Share

Consider the following questions:
What challenged you? What surprised you about your programs?
Could your program have been more streamlined? Have you used too many blocks?
Is there a more efficient way of building your program?
How could your program be used in real-world scenarios?

CONTINUE

Explore text-based programming solutions for this lesson and compare these solutions using different programming languages.

Roaming Vehicles

Design an autonomous car that is safe enough to drive on the streets.

Learning Objectives

Students will:

Make appropriate use of data structures such as lists, tables and arrays
Extend Boolean logic (such as AND, OR and NOT) and some of its uses in circuits and programming
Use the Variable Block to store information
Use the Array Operations Block

Vocabulary

Input, output, pseudocode, variable, Boolean logic, array

Grades

6-8

Subjects

Engineering, STEM, Coding

Duration

45-90 Minutes

Difficulty

Advanced

Standards

NGSS
MS-ETS1-1. / MS-ETS1-2. / MS-ETS1-3. /MS-ETS1-4

CSTA
2-A-2-1 / 2-A-7-2 / 2-A-7-3 / 2-A-7-4 / 2-A-5-5 / 2-A-5-6 / 2-A-5-7 / 2-A-4-8 / 2-A-3-9 / 2-A-6-10 / 2-C-7-11 / 2-C-4-12 / 2-D-5-16 / 2-I-1-20

Materials Needed

LEGO® MINDSTORMS Education EV3 core set
LEGO MINDSTORMS EV3 Software or Programming app
ROBOTC software (optional)

Roaming Vehicles

CONNECT (5 minutes)

This lesson will focus on arrays and how they can be used to control the actions of the students' wheeled robots. Students will discover what arrays are, how they work, and why they are important in computer programming. They will also learn how to incorporate an array into their program.
Ignite a classroom discussion around the following questions:

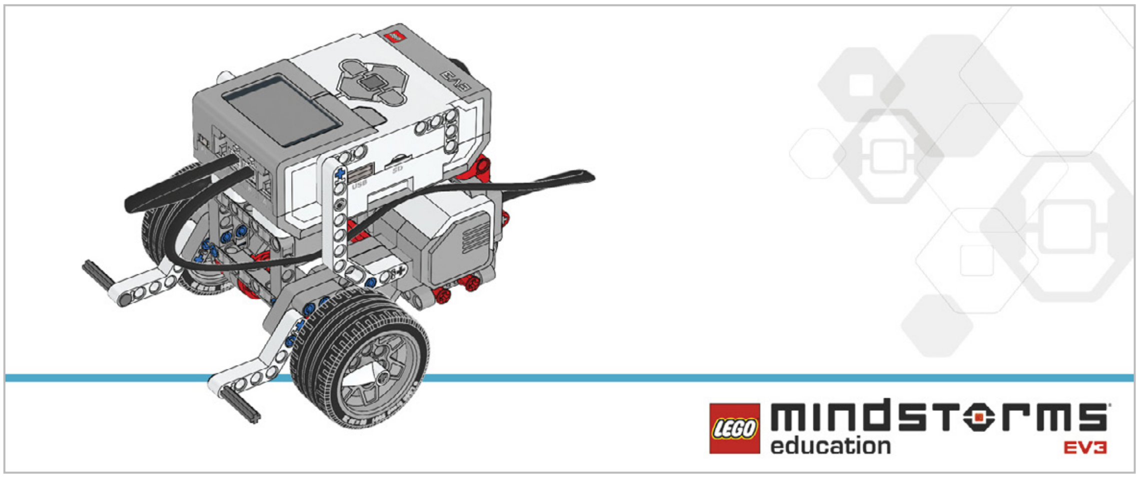
- How do autonomous cars work?
- What would it take to ensure that autonomous cars are safe?
- What types of movements do autonomous cars need to perform?
- Relate this programming task to programming turn by turn directions from a GPS device. Ask the students to discuss, in groups, how they could program turn by turn directions into their robot to make it move around a course.

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their ideas. Encourage them to document their thoughts using text, videos, images, sketchnotes, or another creative medium.

CONSTRUCT (15 to 30 minutes)

Build

Students will construct the Robot Educator base model.



Have the students perform the following building check before they program their robots:

- Are the wires correctly connected from the motors to ports B and C?
- Are the wheels correctly installed?
- Are the wheels rotating freely?

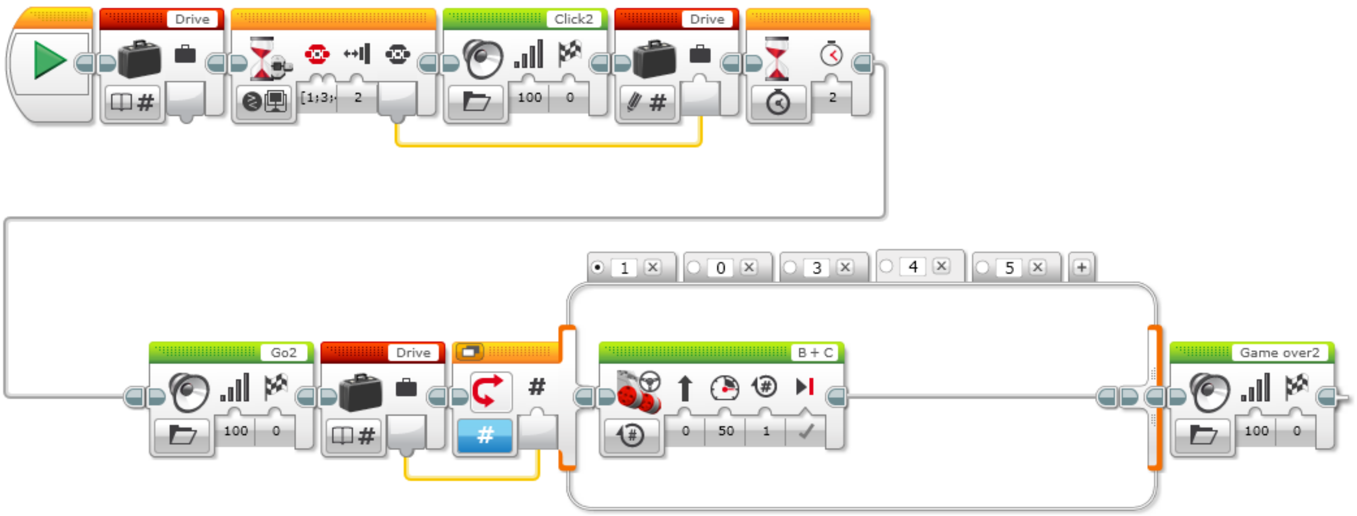
Program

Explain to the students that they are going to program their robot to move according to a recorded set of instructions given to it through the buttons on the EV3 Brick according to these parameters:

- Up Button bumped, the robot moves forward 30 cm
- Down Button bumped, the robot moves backward 30 cm
- Left Button bumped, the robot turns 90 degrees left
- Right Button bumped, the robot turns 90 degrees right

Their program should be divided into two sections: the first is to gather the sequence (e.g., forward, forward, left, forward, right), and the second is to make to robot move according to the sequence. To start with, they should limit the program to recording one item in the array (an array composed of one item is a variable).
Allow the students to select the tool(s) they find most appropriate for capturing and sharing their pseudocode. Encourage them to use text, videos, images, sketchnotes, or another creative medium.

POSSIBLE SOLUTION
FILENAME: CODING-08.EV3 (Tab:1)



RECORDING ONE ACTION TO MAKE THE ROBOT MOVE
1. Start the program.
2. Create a Variable Block called “Drive”.
3. Wait for a Brick Button to be bumped.
4. Play sound “Click 2”.
5. Record the numerical value of the pressed button in the variable “Drive”.
6. Wait for 2 seconds.
7. Play sound “G02”.
8. Read the number stored in variable “Drive” and send the value to a switch.
9. Numeric switch:
   a. If Drive = 1, curve turn the robot left.
   b. If Drive = 3, curve turn the robot right.
   c. If Drive = 4, move the robot straight forward for 2 rotations of the wheels.
   d. If Drive = 5, move the robot straight backward for 2 rotations of the wheels.
10. Play sound “Game Over 2”.

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Note

Refer students to the Robot Educator Tutorials for further assistance.

In the EV3 Software :

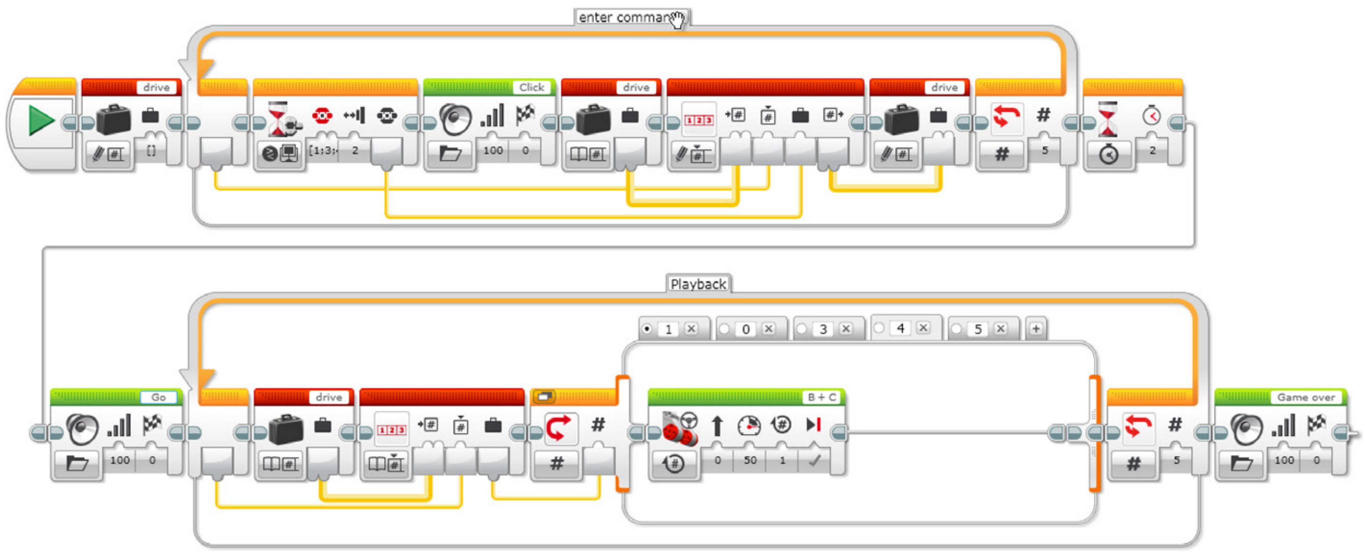
Robot Educator > Beyond Basics > Arrays
Robot Educator > Tools > My Blocks
Robot Educator > Beyond Basics > Loop
Robot Educator > Beyond Basics > Switch
Robot Educator > Beyond Basics > Variables
Robot Educator > Basics > Straight Move
Robot Educator > Basics > Curved Move

The Array Operations Block is used to store a sequence of data. It is often described as a table consisting of one row with multiple columns.

CONTEMPLATE (35 minutes)

Ask the students to improve their program by using an array to record five actions for their robot.

POSSIBLE SOLUTION
FILENAME: CODING-08.EV3 (Tab:2)

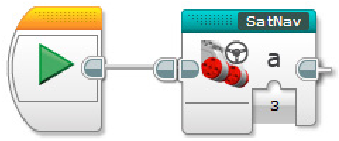


Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

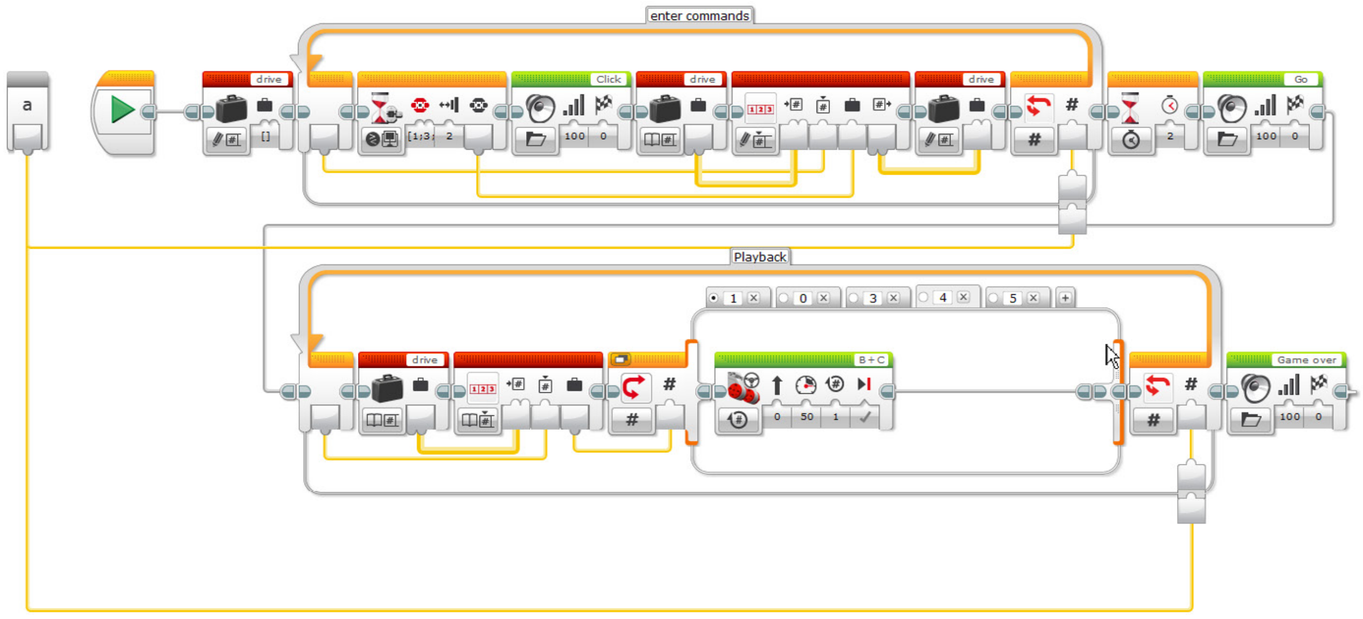
Differentiation Option

Demonstrate to the students how to create a My Block and then add a parameter to it.
Notice that a new tab is added to the Programming Palette showing what is contained within the My Block. To complete the My Block, the user needs to connect the “a” parameter to two inputs within the program. In our program, this will be the two Loop Blocks. This allows data to be directly entered into the My Block, rather than the program contained within it.
Point out that this allows students to easily enter their desired number of steps. Make sure that the students understand that adding a parameter in the My Block Builder allows them to enter data into the My Block that is created.

POSSIBLE SOLUTION
FILENAME: CODING-08.EV3 (Tab:2)



POSSIBLE SOLUTION
FILENAME: CODING-08.EV3 (Tab: SATNAV)



Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

Share

Allow the students to select the tool(s) they find most appropriate for capturing and sharing their creations, unique thinking, and learning process. Encourage them to use text, videos, images, sketchnotes, or another creative medium.
Bring the students together to share their programming successes. Ask how they could have improved their programs.
Does creating a My Block with a parameter make it easier for the students to program the right number of steps?
Ask the students to consider alternative solutions. Can they come up with a new way of programming the robot to move around the room?

Assessment Opportunity
Specific rubrics for assessing computational think can be found under 'Assessment'.

CONTINUE (45 minutes)

Using text-based programming

Have the students explore text-based programming solutions so they can compare different programming languages.

IMPORTANT
The following is a possible solution using the text-based programming language ROBOTC. You may choose to use any other LEGO MINDSTORMS Education EV3 compatible text-based programming languages.

LEGO Education has no ownership of the ROBOTC platform and does not provide any support or guarantee of the quality of the user experience and technology used. All required set up information is provided by ROBOTC at http://www.robotc.net/. We recommend always to reinstall the official LEGO MINDSTORMS EV3 Brick firmware when you finish using other programming languages.

POSSIBLE SOLUTION
FILENAME: CODING-08_1.C

```
#pragma config(StandardModel, "EV3 _ REMBOT")
/*
Use the buttons on the EV3 brick to program the robot to move around.
5 commands can be entered into the EV3 brick. ( Left Button = 1, Right
Button = 3, Up Button = 4, Down Button = 5)
*/
int drive[5];
task main()
{
  for(int i = 0; i < 5; i++) //i = i + 1
```

```
{
     while(getButtonPress(buttonAny) == 0)
     {
            //Wait for Any Button to be pressed.
     }
     if(getButtonPress(buttonLeft) == 1)          drive[i] = 1;
     else if(getButtonPress(buttonRight) == 1)   drive[i] = 3;
     else if(getButtonPress(buttonUp) == 1)      drive[i] = 4;
     else if(getButtonPress(buttonDown) == 1)    drive[i] = 5;
     playSoundFile("Click");
     while(bSoundActive)
     {
            sleep(10);
     }
     while(getButtonPress(buttonAny) == 1)
     {
            //Wait for All Buttons to be released.
            sleep(10);
     }
}
sleep(2000);
playSoundFile("Go");
while(bSoundActive)
{
     sleep(10);
}
for(int i = 0; i < 5; i++)
{
     if(drive[i] == 1)
     {
            //Turn Left Code.
            moveMotorTarget(motorC, 360, 50);
            waitUntilMotorStop(motorC);
     }
     else if(drive[i] == 3)
     {
            //Turn Right Code.
            moveMotorTarget(motorB, 360, 50);
            waitUntilMotorStop(motorB);
     }
     else if(drive[i] == 4)
     {
            //Forward Code.
            moveMotorTarget(motorB, 360, 50);
            moveMotorTarget(motorC, 360, 50);
            waitUntilMotorStop(motorB);
            waitUntilMotorStop(motorC);
        }
        else if(drive[i] == 5)
        {
            //Backward Code.
            moveMotorTarget(motorB, -360, -50);
            moveMotorTarget(motorC, -360, -50);
            waitUntilMotorStop(motorB);
            waitUntilMotorStop(motorC);
        }
    }

    playSoundFile("Game over");
    while(bSoundActive)
    {
        sleep(10);
    }
}
```

Program solutions for this lesson are available for download at:
http://www.LEGOeducation.com

WHAT IS NEXT?

Autonomous Intersection

Roaming Vehicles

Student Worksheets

Design an autonomous car that is safe enough to drive on the streets.

Student Worksheets

CONNECT

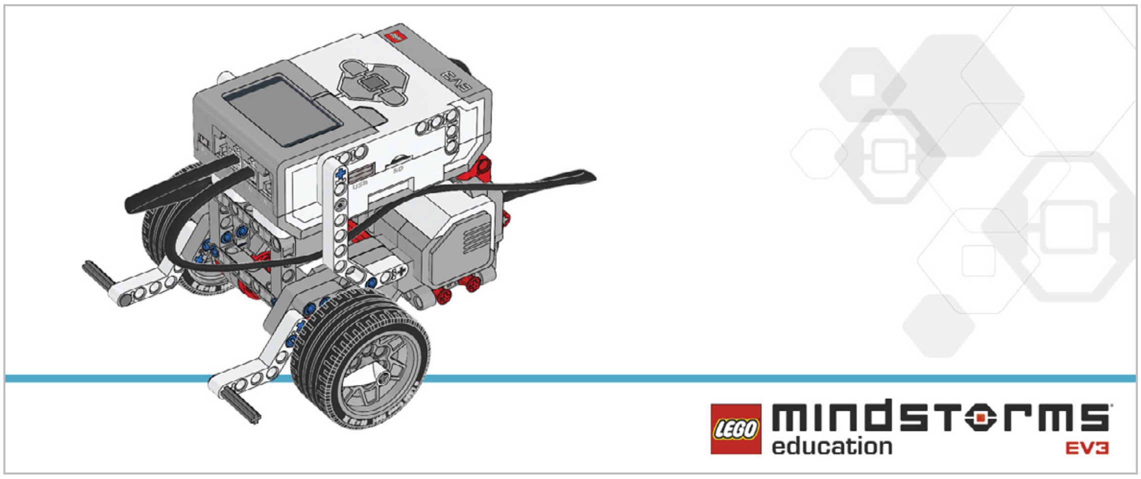
Make sure that you can answer the following questions:
- How do autonomous cars work?
- What would it take ensure that autonomous cars are safe?
- What types of movements do autonomous cars need to perform?

Think about what you have learned, then document it. Describe the problem in your own words. Creatively record your ideas and findings.

CONSTRUCT

Build

Start by constructing this model.

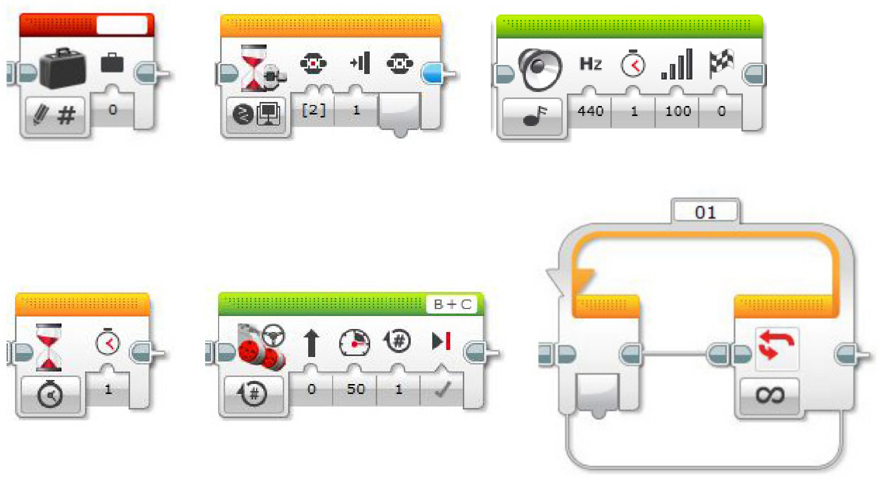


Program

Create a program using arrays so that you can program your robot to move around the room using the EV3 Brick Buttons. The four Brick Buttons can be used as direction controls (i.e., left, right, backward, and forward).
To start, limit your program to one command.
Tip 1: Your program will have two distinct phases:
1. Collecting data
2. Using that data

Tip 2: Two Loop Blocks will be needed for this lesson to allow for the two phases above.
Tip 3: Using the Variable Block often requires a three-step process:
1. Reading the Variable Block
2. Adding information to the block
3. Writing to the block to save the new data

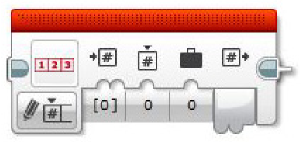
Consider using these blocks in your solution:



Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas and findings.

CONTEMPLATE

Improve your program by using an array to record five actions for your robot.
Consider adding these blocks to your solution:



Think about what you have learned, then document it. Describe your pseudocode for this task. Creatively record your ideas, and findings.

Differentiation option

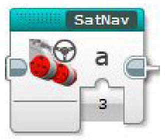
Create a MyBlock to easily edit the number of steps in your program.
To change the number of movement steps from five to something else will require you to edit both Loop Blocks within the program. This can be made much simpler by creating a My Block with a parameter. The My Block will allow the number of loops to be changed easily and clearly.

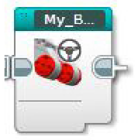
Tip 1: When creating a My Block, highlight the blocks that need to be included, but not the Start Block.

Tip 2: When you need to enter parameters at a later stage, ensure that a parameter has been added to the My Block as shown below. Use the "+" key when creating the block.

Tip 3: The parameter must be joined to the input on the block within the program. In our case, the two loops.

Consider adding these blocks to your solution:





Share

Think about what you have learned, then document it. Creatively record and present your ideas, creations, and findings.

Consider the following questions:
What challenged you? What surprised you during this lesson?
Could your program have been more streamlined? Have you used too many blocks?
Is there a more efficient way of building your program?
How could your program be used in real-world scenarios?

CONTINUE

Explore text-based programming solutions for this lesson and compare these solutions using different programming languages.

Autonomous Intersection

Design an autonomous car that can safely cross an intersection.

Learning Objectives

Students will:

Design, use, and evaluate solutions to a real-world problems and physical systems

Vocabulary

Design brief, prototype, criteria

Grades

6-8

Subjects

Engineering, STEM, Coding

Duration

120+ Minutes

Difficulty

Advanced

Standards

NGSS
MS-ETS1-1. / MS-ETS1-2. / MS-ETS1-3. /MS-ETS1-4

CSTA
2-A-2-1 / 2-A-7-2 / 2-A-7-3 / 2-A-7-4 / 2-A-5-5 / 2-A-5-6 / 2-A-5-7 / 2-A-4-8 / 2-A-3-9 / 2-A-6-10 / 2-C-7-11 / 2-C-4-12 / 2-D-5-16 / 2-I-1-20

Materials Needed

LEGO® MINDSTORMS Education EV3 core set
LEGO MINDSTORMS EV3 Software or Programming app
ROBOTC software (optional)

Autonomous Intersection

CONNECT

Design Brief

The ability to safely cross intersections is a definite challenge for autonomous car builders. Safety on the road is critical and cars need to be aware of danger at all times. They need to detect other cars and traffic lights, and monitor their speed while driving and turning.

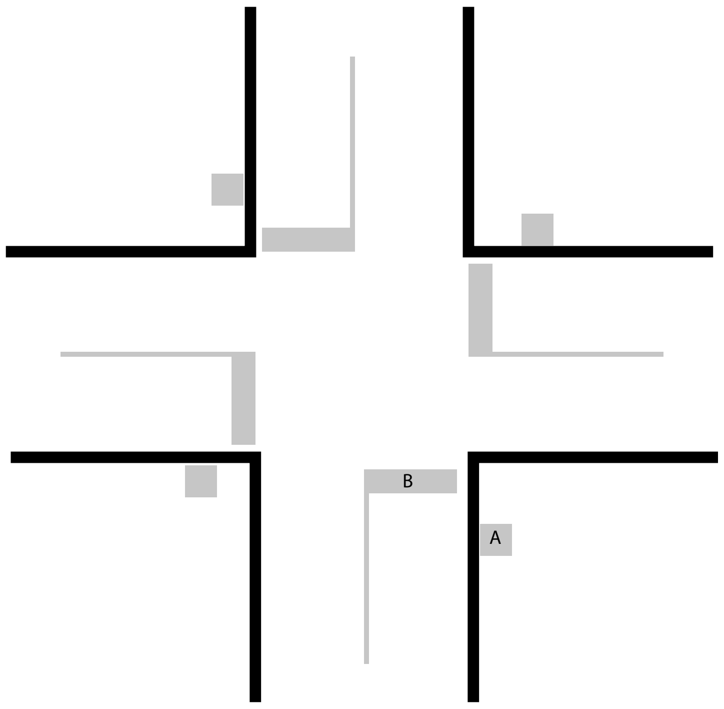
Your challenge is to design a autonomous driving vehicle that can safely cross an intersection. Your solution must be able to:

- Drive to an intersection, properly signal, and turn right
- Repeat the above sequence turning left
- Detect traffic lights and react accordingly to their signals
- Detect other cars and react accordingly

Setup

Design your own classroom setup for this challenge. The setup must provide situational context for the problem the students are trying to solve.

Here is a suggested setup, where A is traffic light position and B is the stop line:



Brainstorm
Have your students look at the design brief and generate multiple possible solutions to the problem.
What features do they intend to include in their build, and in the programming?
Will they need to change the physical design of their robot?
Have them write down all of their thoughts and sketches.

Select the Best Solution
Encourage the students to weigh the pros and cons of each idea they have come up with. They should then decide on a final design to carry forward.

CONSTRUCT
Build and Program a Solution
Have the students build and program a robot that can solve the problem.

CONTEMPLATE
Test and Analyze
Have the students test and evaluate the efficiency of their solution.

Review and Revise
Have the students improve their solution until it meets the design criteria.

Communicate
Encourage the students to share their learning process. Provide them with the opportunity to share their thinking, ideas, and reflections using the documentation tool(s) they have available.

CONTINUE
Using text-based programming

Have the students explore text-based programming solutions so they can compare different programming languages.

LEGOeducation.com/MINDSTORMS

LEGO mindstorms education EV3